PDM Workbench

PDM Workbench Release 3.8 for Aras Innovator

Installation & Administration Manual

Version 2

Copyright

© 2005-2015 T-Systems International GmbH.

All rights reserved. Printed in Germany.

Contact

T-Systems International GmbH
GDC Product Lifecycle Management
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

https://servicenet.t-systems.com/pdm-workbench

☎ +49 (0)40 30600 5544
✉ +49 (0)3915 80125688

mail : *cmi_support@t-systems.com*

Manual History

Version	Date	Version	Date
1.0	April 2005	3.2	March 2012
2.0	November 2006	3.3	October 2012
2.1	November 2007	3.4	April 2013
2.2	September 2008	3.5	October 2013
2.5	September 2010	3.6	April 2014
3.0	October 2011	3.7	October 2014
3.1	February 2012	3.8	April 2015

This edition 3.8 of the manual obsoletes all previous editions.

Your Comments are Welcome

Please feel free to tell us your opinion; we are always interested in improving our publications. Mail your comments to:

T-Systems International GmbH
GDC Product Lifecycle Management
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

mail: *cmi_support@t-systems.com*

Preface

About this Manual

This manual provides installation and configuration information for the PDM Workbench. Before using this guide, be sure you understand:

- the Microsoft Windows operating system
- the administration of the CATIA V5 system
- the administration of the Aras Innovator system

Related Documents

The following manuals contain information about installation, administration, usage and customization of the PDM Workbench:

Manual Title	Version
PDM Workbench Installation & Administration Manual	3.8
PDM Workbench User Manual	3.8

Trademarks

CATIA is a registered trademark of Dassault Systèmes.

Aras is a registered trademark of Aras Corporation.

Names of other products mentioned in this manual are used for identification purpose only and may be trademarks of their companies.

Table of Contents

CHAPTER 1 1

OVERVIEW 1

System and Software Requirements 1
Installation steps 1
Feature license 1

CHAPTER 2 3

ADAPTING CATIA V5 3

Loading PWBCATV5 Software from CD-ROM 3
PWBCATV5 Installation 3
Configuring the installation 3
Silent Installation 11
Parameters 11
Usage 12
Troubleshooting R18 (R17 (64 Bit)) 12
Testing the installation 14
Windows 14
Switching to the PWBSchema file suited for BOM Part Structures 16
PWBSchema modification 16
Setting of Environment Variables 16
Administrative lock for PDM Workbench Preferences 17

CHAPTER 3 19

PDM WORKBENCH DATA MODEL 19

Installation 19

CHAPTER 4 21

PDM WORKBENCH SERVER DLL 21

Copying the DLL 21
Modifying the server configuration file 22

CHAPTER 5 23

CLIENT CUSTOMIZATION 23

Display names 23
Icons 23
Data model definition 24

CHAPTER 6 25

SERVER CONFIGURATION 25

Configuration Variables 25
Configuration Items 25

CHAPTER 7 27

CONFIGURATIONS FOR SPECIFIC FUNCTIONALITIES 27

Standard configuration 27
Exchange map 27
SOAP target URL 27
SOAP client call path environment variable 27
File client call path environment variable 28
PWB Window color 28
Maximum expansion level 28
Read only 28

Reload after update ... 28
Check file versions in structure ... 29
Check CATParts in Visualization mode ... 29
Check CATIA PartNumbers before update ... 29
Allow loading different file versions ... 29
Show out-of-date link info ... 29
Update after create new CATIA file ... 29
Upload changed files at sync failure ... 30
Upload uses Check in Manager ... 30
Required attributes ... 30
Maximum description attribute length ... 30
Show data source NLS values ... 30
Rename existing files in exchange map ... 31
Auto login on session expired ... 31
Session settings ... 31
Language settings ... 31
Date format ... 32
Key attribute ... 32
Class attribute ... 32
Relation attribute ... 32
Relationship attribute ... 32
Left relationship attribute ... 32
Right relationship attribute ... 32
Left relation class attribute ... 32
Right relation class attribute ... 33
Extended relation class attribute ... 33
Last modification date attribute ... 33
Mark superseded nodes ... 33
Check for CAD owner ... 33
Part classes ... 33
Bounding box definition ... 33
TOOLBAR CONFIGURATION ... 34
CUSTOMER-SPECIFIC ENVIRONMENT ... 34
Use case "Query" ... 35
Use case "Update" ... 35
Use case "Load" ... 35
DATA MODEL CONFIGURATION ... 35
BOM Part Structure data model ... 35
Document data model ... 36
BOM PART STRUCTURE CONFIGURATION ... 37
QUERY CONFIGURATION ... 37
QueryOrderByAttribute ... 37
MaxQueryResults ... 37
Default sort criteria for query results ... 37
Display custom query buttons ... 38
Modal dialog ... 38
Check if no query attribute filled ... 38
Display string in query list view ... 38
UPDATE CONFIGURATION ... 39
Create dialogs ... 39
Create version button ... 39
Default create version value ... 40
Show create parts ... 40
Default create parts value ... 41
REVISE CONFIGURATION ... 41
Modifiable attributes ... 41
Expand structure ... 41
NUMBERING ... 42
CadDocNumberAttr ... 42
PartNumberAttr ... 42
INITIAL VERSION STRINGS ... 42
InitialVersionCadDoc ... 42

InitialVersionPart *42*
PROMOTE STATES 43
PromoteSourceStates *43*
PromoteTargetStates *43*
STANDARD PART FUNCTIONALITY 43
StandardPartAdmin *43*
Standard Part functionality for BOM Part Structure data model *43*
Standard Part functionality for Document data model *45*
MANAGEMENT OF CATIA TEMPLATES 47
TemplateFileAdmin *47*
Server configuration *47*
Client configuration *49*
THUMBNAILS 50
DERIVED VIEWABLE FILES 51
ARCHIVES 51
DESIGN TABLES 52
ADD TEMP CONFIGURATION 53
VERSIONING 53
ACCESSIBILITY OF OLD FILE VERSIONS 53
NON-BOM CATPARTS AND CATPRODUCTS 54
CHECK CAD LINKS 54
DISPLAYING PART STRUCTURE INSTANCES AS SEPARATE NODES 55
SUPPORT FOR THE NEW CAD STRUCTURE INSTANCE HANDLING INTRODUCED IN INNOVATOR 9.4 AND 10.0 55
ATTRIBUTE MAPPING CONFIGURATION 55
Standard attribute mapping *55*
User attribute mapping *58*
CATIA user defined attributes *60*
Drawing attribute mapping *61*
Extended attribute mapping functionality *61*
BASIC DRAWING LINK SUPPORT 62
Download related 3D File *62*
Duplicate related drawings *62*
BASIC MULTI-MODEL LINK SUPPORT 63
AUTONAME SUPPORT USING INNOVATOR SEQUENCE ITEMS 63
AUTONAME FUNCTIONALITY CAN USE A SERVER METHOD 64
CONFIGURABLE CATIA COMPONENTS SUPPORT 66
SUPPORT FOR RELATING A NEW CATIA FILE TO AN EXISTING PART 66
"DELETE RELATION" CONTEXT MENU ACTION IN THE PDM STRUCTURE WINDOW 66
THE PE_DELETECADSTRUCTURE SERVER EVENT METHOD CAN BE REMOVED 67
USING CATIA V4 MODELS 67
CHECK FOR CAD DOCUMENT CATIA RELEASE AT PDM UPDATE 68
LOCAL WORKSPACE INFORMATION 68
CONFIGURATION OF BOM PART STRUCTURE 69
POSSIBILITY TO CALL A SERVER METHOD FOR A PDM ITEM 70
PART CAD FILTER 72
Configuration *73*
REVISING AS A DIFFERENT USER ('REVISEAS') 74
Configuration *74*

CHAPTER 8 75

CLIENT SCHEMA FILE CONFIGURATION 75

STRUCTURE OF THE SCHEMA FILE 75
Attributes of the tag "PWBSchema": *75*
NLS Support for Display Names *76*
Configuration settings *76*
"object": 1 - n *79*
"relation": 1 - n *79*
"attribute": 0 - n *79*
"pwbAttribute": 0 - n *79*
"dataSource": 0 - n *79*

PDM ATTRIBUTES AND FORM ATTRIBUTES 80
Description of the Widget Types *81*
Login Form *83*
PDM OBJECTS 84
Description of PDM Objects *84*
Tooltip of PDM Objects *85*
Actions on PDM Objects *85*
Context Actions *86*
Disabling context menu items in the CATIA structure window *88*
PDM Object Forms *88*
PDM RELATIONS 88
Description of PDM Relations *89*
"Relationship" tags *89*
"Left and Right Object" Classes *90*
PDM Relation Forms *90*
DATA SOURCES 91
Data Source "Value" tag *91*
Complete example of using a data source tag: *92*

Table of Figures

PICTURE 1: DIRECTORY STRUCTURE OF THE PDM WORKBENCH INSTALLATION FILES 3
PICTURE 2: WELCOME TO THE INSTALLATION 5
PICTURE 3: LICENSE AGREEMENT 5
PICTURE 4: CHOOSE INSTALLATION SCOPE 6
PICTURE 5: CHOOSE PDM PACKAGE 6
PICTURE 6: CHOOSE PDM PACKAGE (WITH PROPOSAL) 7
PICTURE 7: CHOOSE INSTALL LOCATION 7
PICTURE 8: CHOOSE CATIA V5 INSTALLATION 8
PICTURE 9: CATIA V5 ENVIRONMENT FILE SELECTION 8
PICTURE 10: PWB EXCHANGE DIRECTORY 9
PICTURE 11: LOCATION OF SOAP TARGET URL 9
PICTURE 12: INSTALLATION PROGRESS 10
PICTURE 13: INSTALLATION PROGRESS END 10
PICTURE 14: INSTALLATION FINISHED 11
PICTURE 15: EVENT PROPERTIES 13
PICTURE 16: PDM WORKBENCH TOOLBAR BEFORE THE LOGIN 14
PICTURE 17: THE PDM WORKBENCH TOOLBAR AFTER THE LOGIN 14
PICTURE 18: CATIA V5 GENERAL→GENERAL SETTINGS 15
PICTURE 19: CATIA V5 GENERAL→DOCUMENT SETTINGS 15
PICTURE 20: PDM SESSION CONFIGURATION DIALOG 16
PICTURE 21: PDM WORKBENCH PREFERENCES (ADMINISTRATOR VIEW) 17
PICTURE 22: PDM WORKBENCH PREFERENCES (USER VIEW) 18
PICTURE 23: PDM ARAS INNOVATOR IMPORT UTILITY 19
PICTURE 24: ARAS INNOVATOR SERVER CONFIGURATION VARIABLES 25
PICTURE 25: PWB CONFIGURATION ITEM IN ARAS INNOVATOR 26
PICTURE 26: SAMPLE USEBOMPARTSTRUCTURE CONFIGURATION 35
PICTURE 27: STRUCTURE IN THE BOM PART STRUCTURE DATA MODEL 36
PICTURE 28: STRUCTURE IN THE DOCUMENT DATA MODEL 36
PICTURE 29: SAMPLE BOMPARTSTRUCTURELOCKUNLOCKRELATEDITEM CONFIGURATION 37
PICTURE 30: SAMPLE QUERYORDERBYATTRIBUTE CONFIGURATION 37
PICTURE 31: SAMPLE MAXQUERYRESULTS CONFIGURATION 37
PICTURE 32: QUERY LIST VIEW DIALOG WITH DISPLAY STRING 39
PICTURE 33: SAMPLE SHOWCREATEDIALOGSDURINGUPDATE CONFIGURATION 39
PICTURE 34: UPDATE DIALOG WITH "CREATE NEW FILE VERSIONS AT UPDATE?" 40
PICTURE 35: UPDATE DIALOG WITH "CREATE NEW PARTS AT UPDATE?" 40
PICTURE 36: REVISE DIALOG WITH MODIFIABLE ATTRIBUTES "MAJOR_REV" AND "DESCRIPTION" 41
PICTURE 37: SAMPLE CADDOCNUMBERATTR CONFIGURATION 42
PICTURE 38: SAMPLE PARTNUMBERATTR CONFIGURATION 42
PICTURE 39: SAMPLE INITIALVERSIONCADDOC CONFIGURATION 42
PICTURE 40: SAMPLE INITIALVERSIONPART CONFIGURATION 42
PICTURE 41: SAMPLE PROMOTESOURCESTATES CONFIGURATION 43
PICTURE 42: SAMPLE PROMOTETARGETSTATES CONFIGURATION 43
PICTURE 43: SAMPLE STANDARDPARTADMIN CONFIGURATION 44
PICTURE 44: MAKING CATPARTS IN A LOCAL FOLDER ACCESSIBLE 44
PICTURE 45: DEFINING A CATPART AS A STANDARD PART 45
PICTURE 46: "STANDARD PART" CHECK BOX FOR CAD DOCUMENTS 45
PICTURE 47: "ISSTANDARDPART" USER-DEFINED PROPERTY 46
PICTURE 48: THREE EXAMPLE STANDARD PART CATPARTS 46
PICTURE 49: USER IS LOGGED IN AS ADMINISTRATOR 46
PICTURE 50: THE STANDARD PART ADMINISTRATOR IDENTITY 47
PICTURE 51: TEMPLATE FILE ADMINISTRATOR CONFIGURATION VARIABLE 47
PICTURE 52: THE TEMPLATE FILE ADMINISTRATOR IDENTITY 48
PICTURE 53: PWB CONFIGURATION VARIABLES FOR TEMPLATE CREATION 48
PICTURE 54: CREATE DIALOG CONTAINING "IS TEMPLATE" CHECKBOX 48
PICTURE 55: TEMPLATE FILE CREATION ERROR MESSAGE 49
PICTURE 56: SAMPLE CREATETHUMBNAILSFROMTYPES CONFIGURATION 50

PICTURE 57: SAMPLE DERIVED VIEWABLE FILES CONFIGURATION 51
PICTURE 58: SAMPLE USEARCHIVES CONFIGURATION 51
PICTURE 59: CLASS STRUCTURE FOR TYPE 'CAD' – ARCHIVE ADDED 52
PICTURE 60: SAMPLE USEDESIGNTABLES CONFIGURATION 52
PICTURE 61: ITEM TYPE "PART" 53
PICTURE 62: ITEM TYPE "CAD" 54
PICTURE 63: SAMPLE USENONBOM3DCADDOCS CONFIGURATION 54
PICTURE 64: SAMPLE CADLINKCHECKQUERYATTRIBUTE CONFIGURATION 54
PICTURE 65: SAMPLE USESEPARATERELATIONSFORINSTANCES CONFIGURATION 55
PICTURE 66: EVERY PART INSTANCE IS SHOWN AS SEPARATE NODE 55
PICTURE 67: SAMPLE USECADINSTANCE CONFIGURATION 55
PICTURE 68: STANDARD ATTRIBUTES IN THE "PROPERTIES" DIALOG 56
PICTURE 69: CONFIGURATION OF STANDARD ATTRIBUTES IN ARAS INNOVATOR 57
PICTURE 70: STANDARD ATTRIBUTES IN THE "PROPERTIES" DIALOG OF THE PDM NODE 57
PICTURE 71: STANDARD ATTRIBUTES IN ARAS INNOVATOR WINDOW 58
PICTURE 72: CONFIGURATION OF USER-DEFINED ATTRIBUTES IN ARAS INNOVATOR 58
PICTURE 73: USER-DEFINED ATTRIBUTES IN THE "PROPERTIES" DIALOG OF THE PDM NODE 59
PICTURE 74: USER-DEFINED ATTRIBUTES IN ARAS INNOVATOR WINDOW 59
PICTURE 75: USER-DEFINED ATTRIBUTES IN THE "PROPERTIES" DIALOG 60
PICTURE 76: CONFIGURATION OF DRAWING ATTRIBUTES IN ARAS INNOVATOR 61
PICTURE 77: EXAMPLE PWB CONFIGURATION ATTRIBUTE MAPPING 61
PICTURE 78: SAMPLE SEQUENCE ITEM 63
PICTURE 79: SEQUENCE ITEMS USED IN EXAMPLE 63
PICTURE 80: THE PE_DELETECADSTRUCTURE SERVER EVENT METHOD 67
PICTURE 81: SAMPLE CADSTRUCTRELSDELETEDAFTERVERSION CONFIGURATION 67
PICTURE 82: CLASS STRUCTURE FOR TYPE 'CAD' – CATIAV4MODEL ADDED 67
PICTURE 83: BOM CONFIGURATION MANAGEMENT TYPE OVERVIEW 69
PICTURE 84: BOM CONFIGURATION MANAGEMENT TYPES IN ARAS INNOVATOR 70
PICTURE 85: EXAMPLE REP TYPE ATTRIBUTE 73
PICTURE 86: EXAMPLE REP TYPE VALUE LIST 73
PICTURE 87: SAMPLE PART CAD FILTER CONFIGURATION 73
PICTURE 88: DIFFERENT REP TYPE VALUES ON CAD DOCUMENTS IN A STRUCTURE 74
PICTURE 89: SAMPLE 'REVISEAS' SETTING 74
PICTURE 90: SINGLE LINE EDITOR WIDGET, UPDATE MODE 81
PICTURE 91: SINGLE LINE EDITOR WIDGET, OUTPUT MODE 82
PICTURE 92: MULTI LINE EDITOR WIDGET, UPDATE MODE 82
PICTURE 93: COMBO BOX WIDGET, SELECT MODE 82
PICTURE 94: SINGLE CHECK BOX WIDGET, SELECT MODE 82
PICTURE 95: CHECK BOXES WIDGET, SELECT MODE 82
PICTURE 96: RADIO BUTTONS WIDGET, SELECT MODE 83
PICTURE 97: SINGLE SELECTOR LIST WIDGET, SELECT MODE 83
PICTURE 98: PDM NODE IN PWB WINDOW 85
PICTURE 99: TOOLTIP OF PDM NODE IN PWB WINDOW 85
PICTURE 100: SELECT PDM OBJECT TYPE IN "PDM QUERY" DIALOG 86
PICTURE 101: CONTEXT ACTIONS FOR THE TYPE /PART/ASSEMBLY 87
PICTURE 102: RELATION ICON WITH RELATIONSHIP AND DESCRIPTION ATTRIBUTE 90

CHAPTER 1

Overview

This chapter provides basic information about the installation of the *PDM Workbench*.

System and Software Requirements

Server Installation of Aras Innovator 10.0 or 11.0 on the following operation systems:

- Windows Server 2008, Windows Server 2012

On the CATIA client computers .NET 4 for Innovator 10.0 and 11.0 and MSXML4 SP3 have to be installed.

CATIA V5 Client Version CATIA V5 Client V5-6R2013, CATIA V5 Client V5-6R2014, and CATIA V5 Client V5-6R2015 on the following operation systems:

- Windows 7 (64 Bit)

Important notice:

CATIA V5-6R2014 SP2 has been retracted by Dassault Systèmes and is not supported. Please use SP3 instead.

Installation steps

This section describes which PDM Workbench modules (client and server) need to be installed.

On the client and the server two steps need to be performed each:

- Client installation: CATIA V5 Add-in (chapter 2)
- Client installation: License Manager (For the installation of “licman20” please refer to the *Licman 2.0 Installation Manual*.)
- Server installation: PDM Workbench data model and server methods (chapter 3)
- Server installation: PDM Workbench server DLL (chapter 4)

Feature license

In addition to these steps an additional feature license for the connector needs to be requested from Aras Corporation.

Feature licenses are requested by sending an email to *licenses@aras.com* which includes the requested feature (“T-Systems.CATIAV5Connector”), the account (company) name, and the contact person’s name.

The Aras license team will process this request and issue the feature license activation key once the request is validated.

A list of all available feature licenses can be accessed from within Aras Innovator, using the *Tools → Admin → Licenses → View Feature Tree* command.

Trial license requests are submitted to *licenses@aras.com* the same as regular license requests, by the requesting company.

CHAPTER 2

Adapting CATIA V5

The PWBCATV5 module provided by T-Systems International GmbH extends the CATIA V5 functionality to communicate with the Aras Innovator PDM system.

You should perform the following steps with your CATIA system administrator.

The **PWBCATV5_Rxx_xxVxx** module includes all of the supported platform data in a compressed file. Thus, you should choose an installation location for all CATIA V5 clients.

In the following example sections it is supposed that the software will be installed within the directory **C:\Program Files\T-Systems\PWBCATV5_Rxx_xxVxx_Aras_xx** on Windows but you can surely choose any other destination for the module.

Within the installation you will need to supply the PDM specific installation package. The file name follows the naming convention **PWBCATV5_xxVxx_Aras_xx**. Where the substring _xxVxx_ matches the corresponding substring of the CATIA module name.

Loading PWBCATV5 Software from CD-ROM

Windows 7

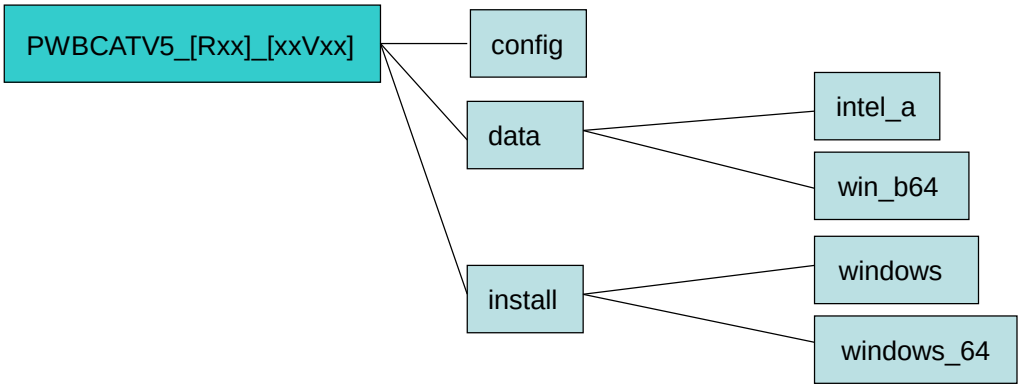
Use the Windows Explorer to locate the **D:\pwbcatv5\PWBCATV5_[Rxx]_[xxVxx].zip** file on the CD. Extract the content of the archive file to a temporary installation location.

PWBCATV5 Installation

After you have successfully transferred the installation files to your installation host; the following steps will install the files and configure your installation.

Configuring the installation

The **PWBCATV5_[Rxx]_[xxVxx]** Installation Directory has the following structure:



Picture 1: Directory structure of the PDM Workbench installation files

The **config** directory contains readme files and special files needed by the installer or the installed program.

The **data** directory contains the binary distributions for the PWBCATV5 module for the supported operating system mnemonics.

The supported operation systems and their mnemonics are:

Windows 7 (32 Bit)	intel_a
Windows 7 (64 Bit)	win_b64

The **install** directory contains the sub directories **windows**, and **windows_64** with all necessary data for the installer program.

Windows 7 (64 Bit)

On **Windows 7 (64 Bit)** use the Windows Explorer to run the **setup.exe** in the directory **PWBCATV5_[Rxx]_[xxVxx]\install\windows** of the installation package if you have installed the 32 Bit version of CATIA V5.

On **Windows 7 (64 Bit)** use the Windows Explorer to run the **setup.exe** in the directory **PWBCATV5_[Rxx]_[xxVxx]\install\windows_64** of the installation package if you have installed the 64 Bit version of CATIA V5.

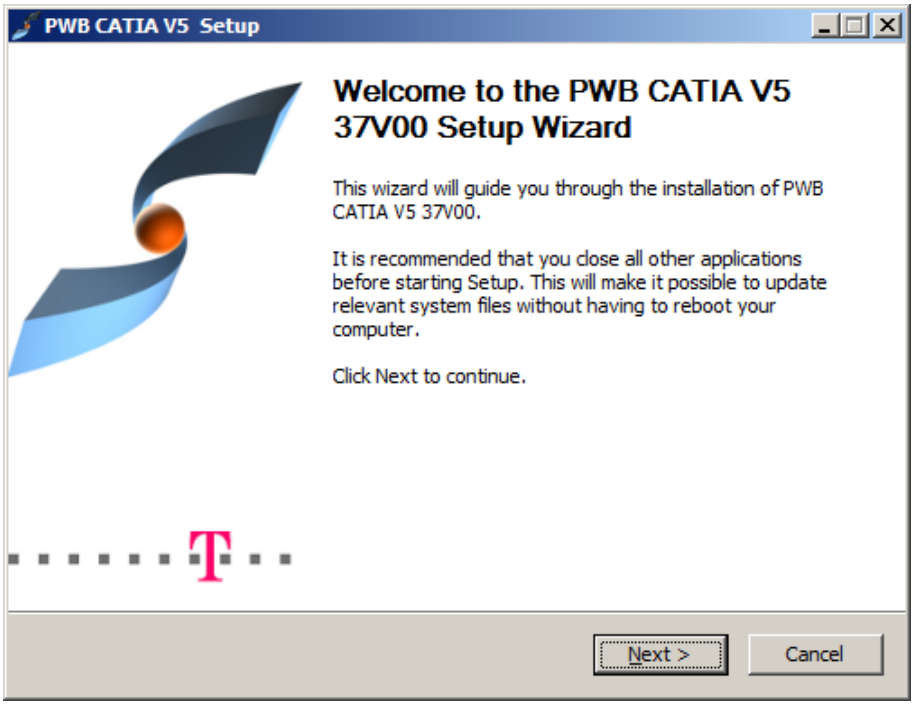
On **Windows Vista**/**Windows 7** the User Account Control (UAC) will be triggered and you will have to agree that the setup program may make changes to the computer. The installer is signed with a "**T-Systems International GmbH**" certificate to ensure its integrity and source.

The setup will **NOT** modify the native installation of CATIA V5.

The licman20 license manager has to be installed on the CATIA V5 client host. For the installation of the license manager please refer to the *Licman 2.0 Installation Manual*.

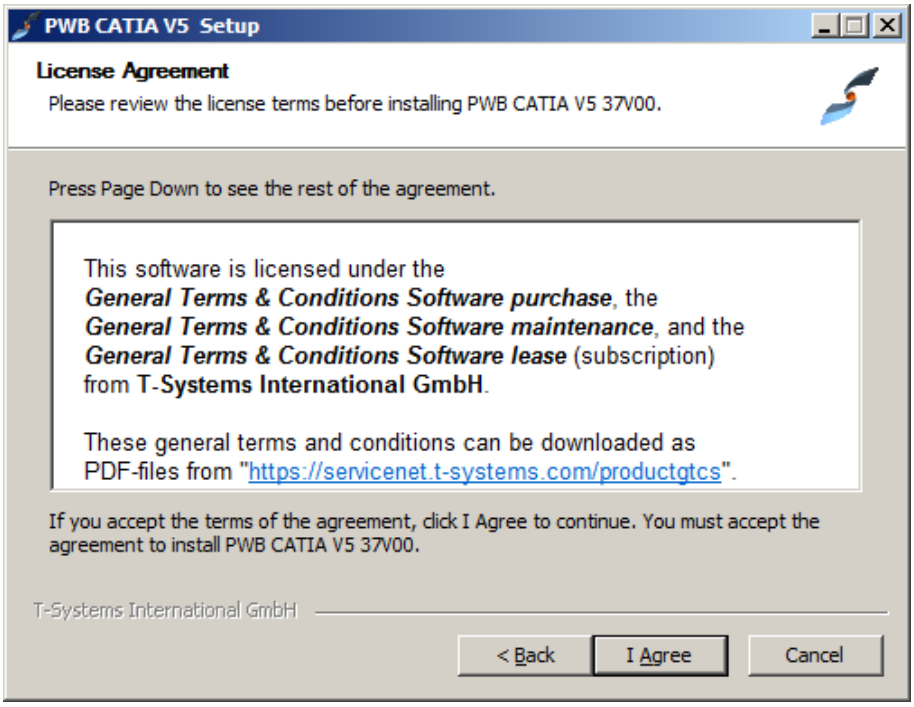
In the following the setup is shown step-by-step.

Installation process:


PWB CATIA V5 Setup
Welcome to the PWB CATIA V5 37V00 Setup Wizard
This wizard will guide you through the installation of PWB CATIA V5 37V00.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

Picture 2: Welcome to the Installation

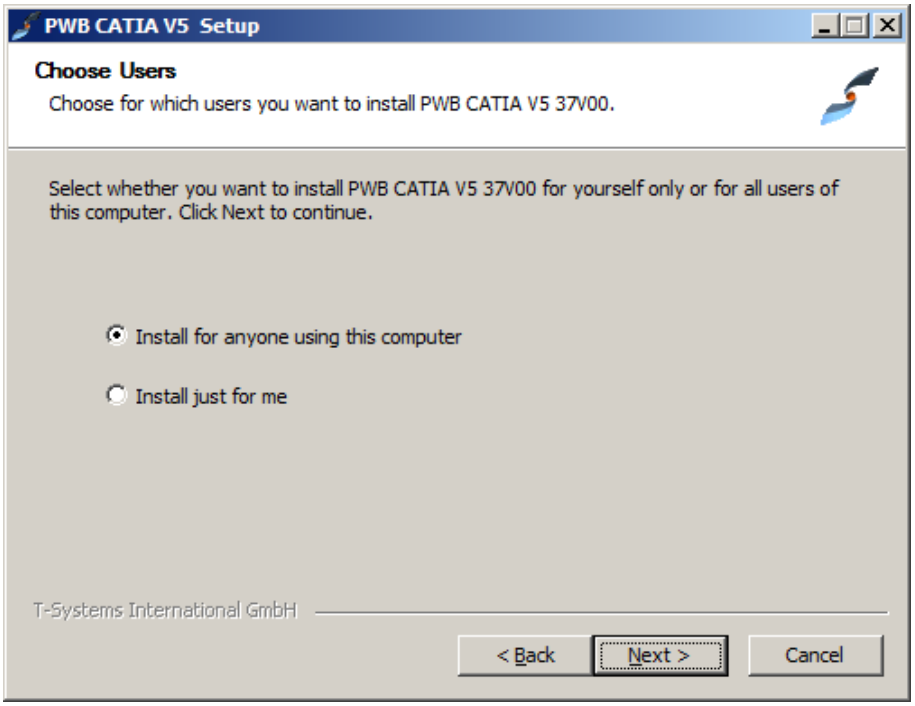
The installer software asks to approve the license terms (see *Picture 3: License Agreement*).


PWB CATIA V5 Setup
License Agreement
Please review the license terms before installing PWB CATIA V5 37V00.
Press Page Down to see the rest of the agreement.
This software is licensed under the
General Terms & Conditions Software purchase, the
General Terms & Conditions Software maintenance, and the
General Terms & Conditions Software lease (subscription)
from T-Systems International GmbH.
These general terms and conditions can be downloaded as PDF-files from "https://servicenet.t-systems.com/productgtcs".
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install PWB CATIA V5 37V00.
T-Systems International GmbH
< Back
I Agree
Cancel

Picture 3: License Agreement

The installer software asks for the following input: **User scope**.

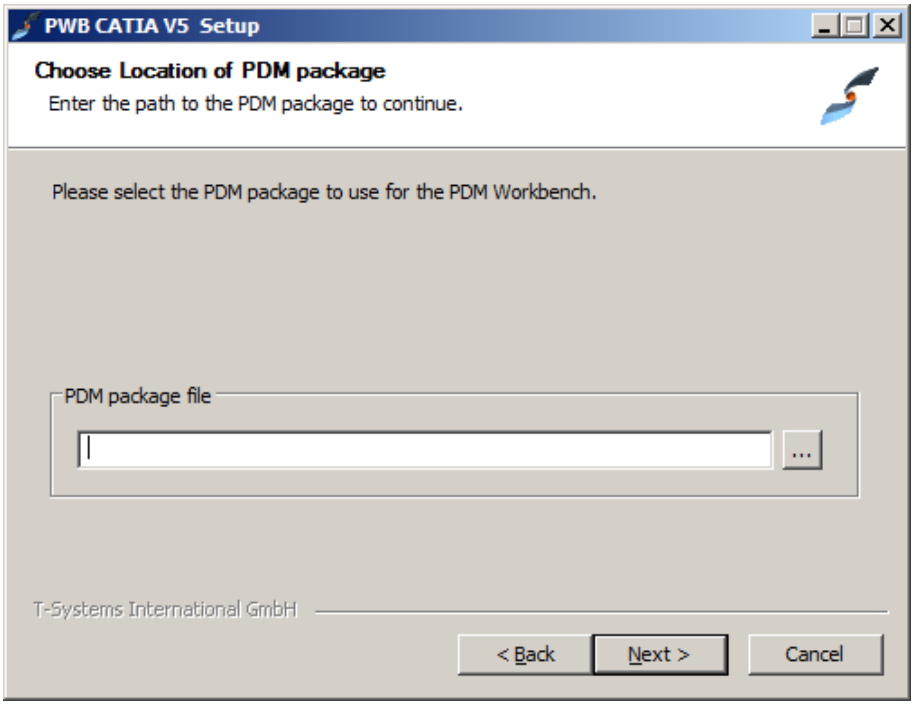
Next the installer will ask you to define the scope of the installation (see *Picture 4: Choose installation scope*). You can choose between an installation for anyone using the computer or just for the current user.


PWB CATIA V5 Setup
Choose Users
Choose for which users you want to install PWB CATIA V5 37V00.
Select whether you want to install PWB CATIA V5 37V00 for yourself only or for all users of this computer. Click Next to continue.
Install for anyone using this computer
Install just for me
T-Systems International GmbH
< Back
Next >
Cancel

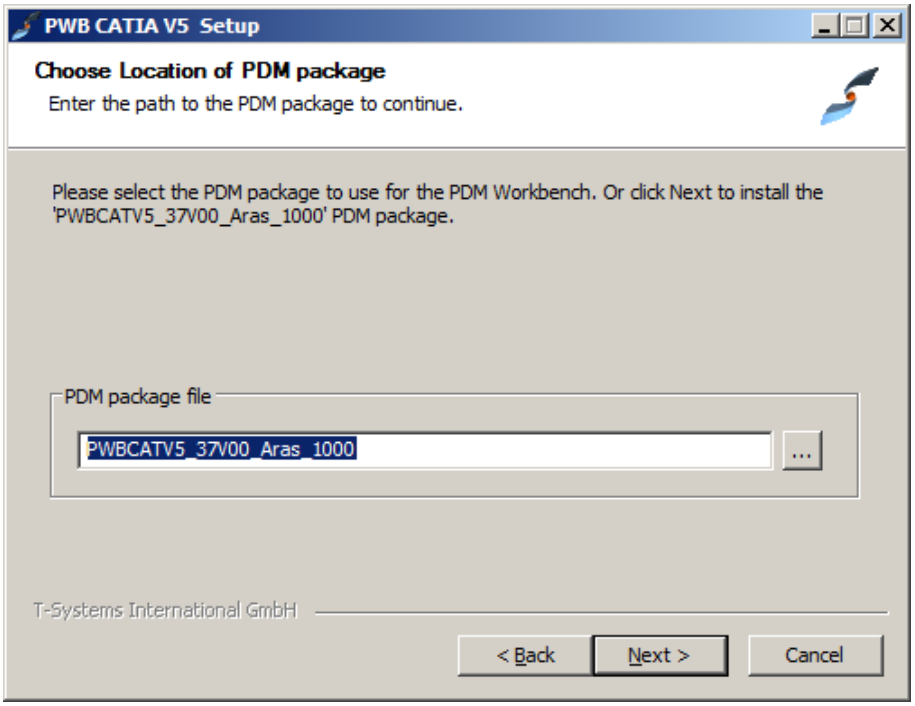
Picture 4: Choose installation scope

The installer software asks for the following input: **Location of the PDM package.**

The installer asks for the location of the PDM package to use (see *Picture 5: Choose PDM package*). If an PDM package has previously been unpacked within the installer it will be offered to install this package directly (see *Picture 6: Choose PDM package (with proposal)*).


PWB CATIA V5 Setup
Choose Location of PDM package
Enter the path to the PDM package to continue.
Please select the PDM package to use for the PDM Workbench.
PDM package file
...
T-Systems International GmbH
< Back
Next >
Cancel

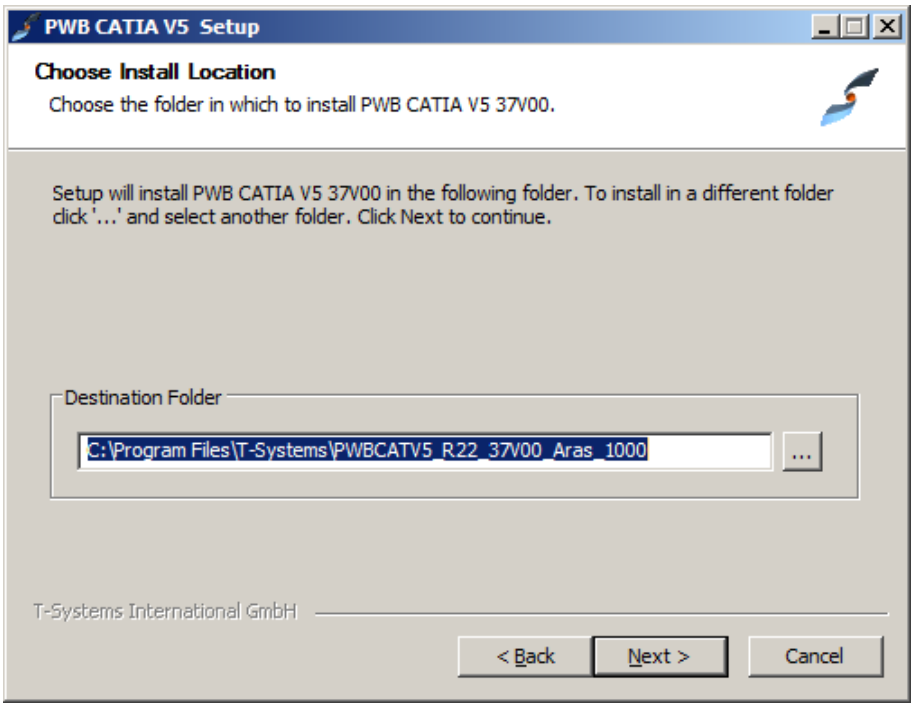
Picture 5: Choose PDM package


PWB CATIA V5 Setup
Choose Location of PDM package
Enter the path to the PDM package to continue.
Please select the PDM package to use for the PDM Workbench. Or click Next to install the 'PWBCATV5_37V00_Aras_1000' PDM package.
PDM package file
PWBCATV5_37V00_Aras_1000
...
T-Systems International GmbH
< Back
Next >
Cancel

Picture 6: Choose PDM package (with proposal)

The installer software asks for the following input: **Installation directory.**

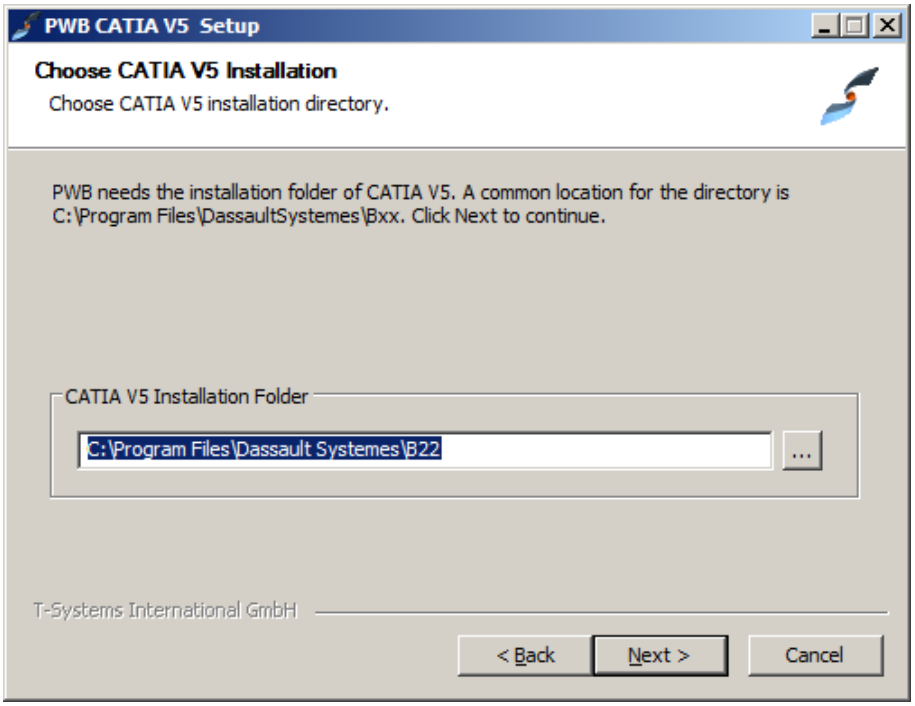
Next the installer will ask you for the target directory for the installation. You can use the given standard location or choose any other location (see *Picture 7: Choose Install Location*). The chosen folder must be empty or not existent.


PWB CATIA V5 Setup
Choose Install Location
Choose the folder in which to install PWB CATIA V5 37V00.
Setup will install PWB CATIA V5 37V00 in the following folder. To install in a different folder click '...' and select another folder. Click Next to continue.
Destination Folder
C:\Program Files\T-Systems\PWBCATV5_R22_37V00_Aras_1000
...
T-Systems International GmbH
< Back
Next >
Cancel

Picture 7: Choose Install Location

The installer software asks for the following input: **CATIA installation directory**

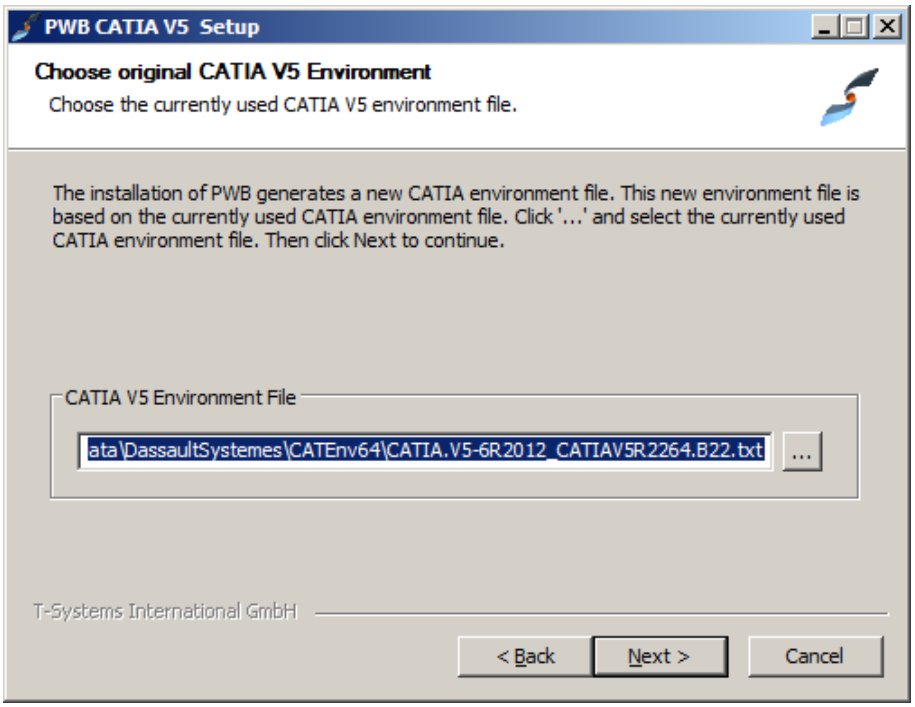
The installation path of the CATIA to use needs to be specified (see *Picture 8: Choose CATIA V5* Installation).


PWB CATIA V5 Setup
Choose CATIA V5 Installation
Choose CATIA V5 installation directory.
PWB needs the installation folder of CATIA V5. A common location for the directory is C:\Program Files\DassaultSystemes\Bxx. Click Next to continue.
CATIA V5 Installation Folder
C:\Program Files\Dassault Systemes\B22
...
T-Systems International GmbH
< Back
Next >
Cancel

Picture 8: Choose CATIA V5 Installation

Afterwards you will be asked for your CATIA V5 environment file (see *Picture 9: CATIA V5 Environment File Selection*).

The installation of PDM Workbench generates a new CATIA V5 environment file. This new environment file is based on the currently used CATIA V5 environment file.


PWB CATIA V5 Setup
Choose original CATIA V5 Environment
Choose the currently used CATIA V5 environment file.
The installation of PWB generates a new CATIA environment file. This new environment file is based on the currently used CATIA environment file. Click '...' and select the currently used CATIA environment file. Then click Next to continue.
CATIA V5 Environment File
ata\DassaultSystemes\CATEnv64\CATIA.V5-6R2012_CATIAV5R2264.B22.txt
...
T-Systems International GmbH
< Back
Next >
Cancel

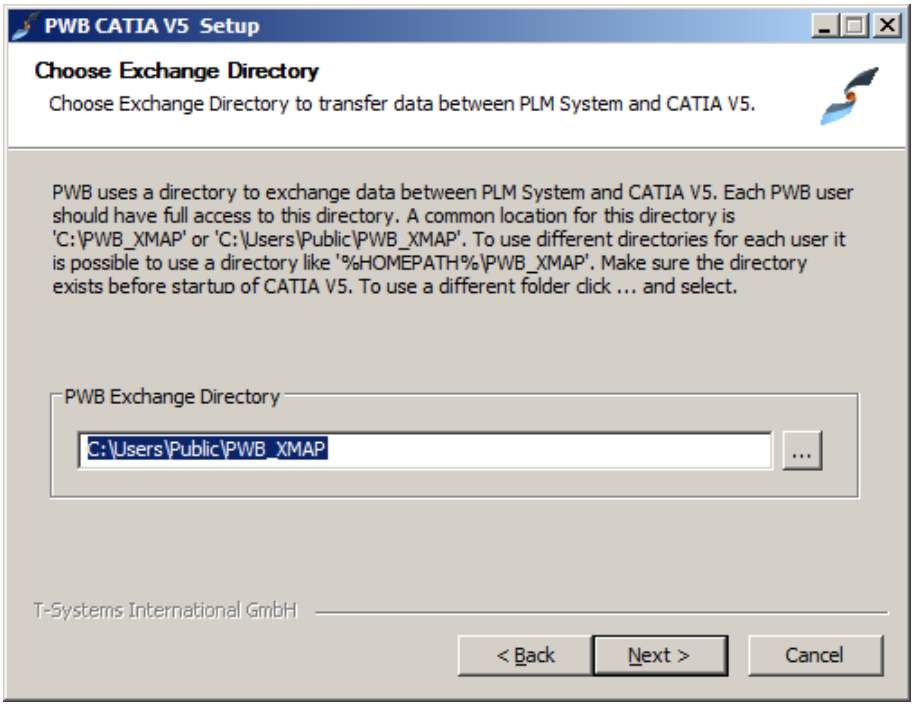
Picture 9: CATIA V5 Environment File Selection

The installer software asks for the following input: **PWB Exchange Directory**.

The PDM Workbench needs a temporary directory to perform the file transfer between CATIA and the PDM system. Make sure this directory exists for every PDM Workbench user on the CATIA client machine.

You can either use the standard location or choose any other location (see *Picture 10: PWB Exchange Directory*).

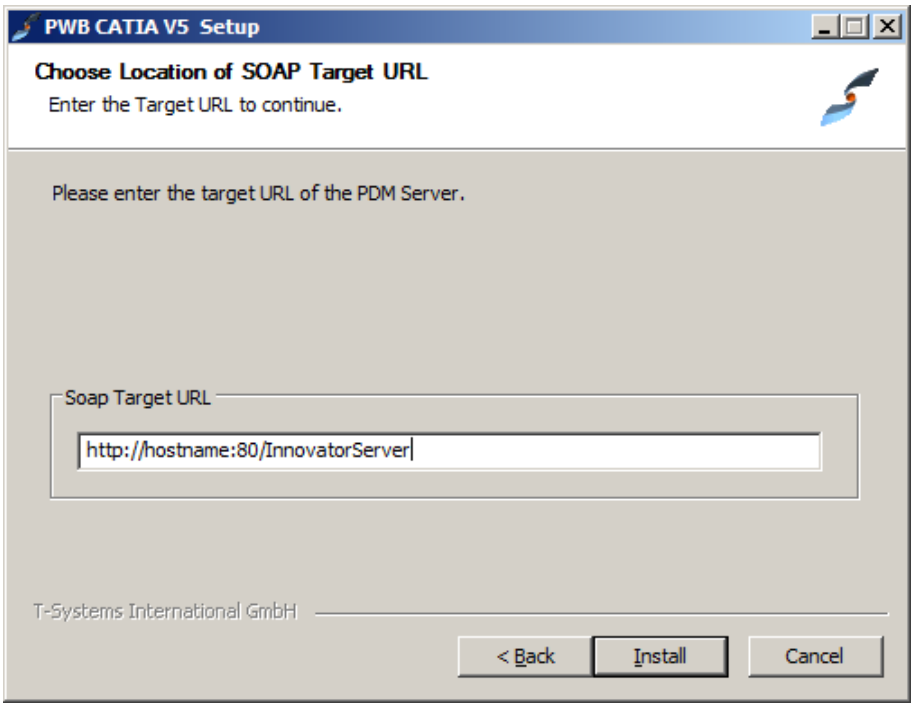
If it is planned to run more than one CATIA session at a time each session must use its own PWB Exchange Directory!


PWB CATIA V5 Setup
Choose Exchange Directory
Choose Exchange Directory to transfer data between PLM System and CATIA V5.
PWB uses a directory to exchange data between PLM System and CATIA V5. Each PWB user should have full access to this directory. A common location for this directory is 'C:\PWB_XMAP' or 'C:\Users\Public\PWB_XMAP'. To use different directories for each user it is possible to use a directory like '%HOMEPATH%\PWB_XMAP'. Make sure the directory exists before startup of CATIA V5. To use a different folder click ... and select.
PWB Exchange Directory
C:\Users\Public\PWB_XMAP
...
T-Systems International GmbH
< Back
Next >
Cancel

Picture 10: PWB Exchange Directory

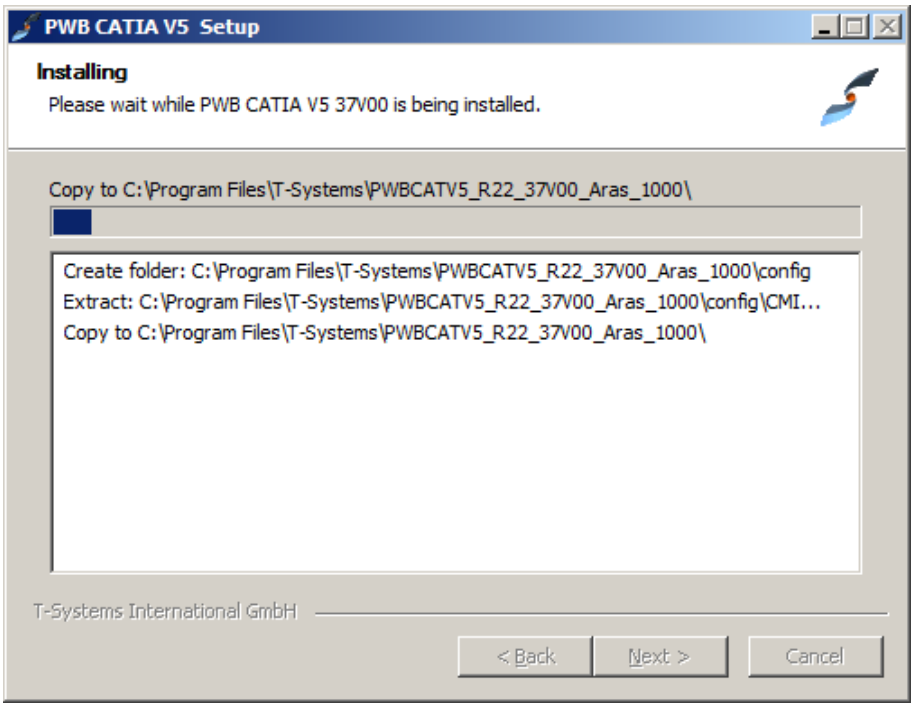
Finally you have to define the so called “Soap Target URL” for the PDM Server (see *Picture 11: Location of SOAP Target URL*).

This URL defines the host and port on which the PDM Server is reachable.

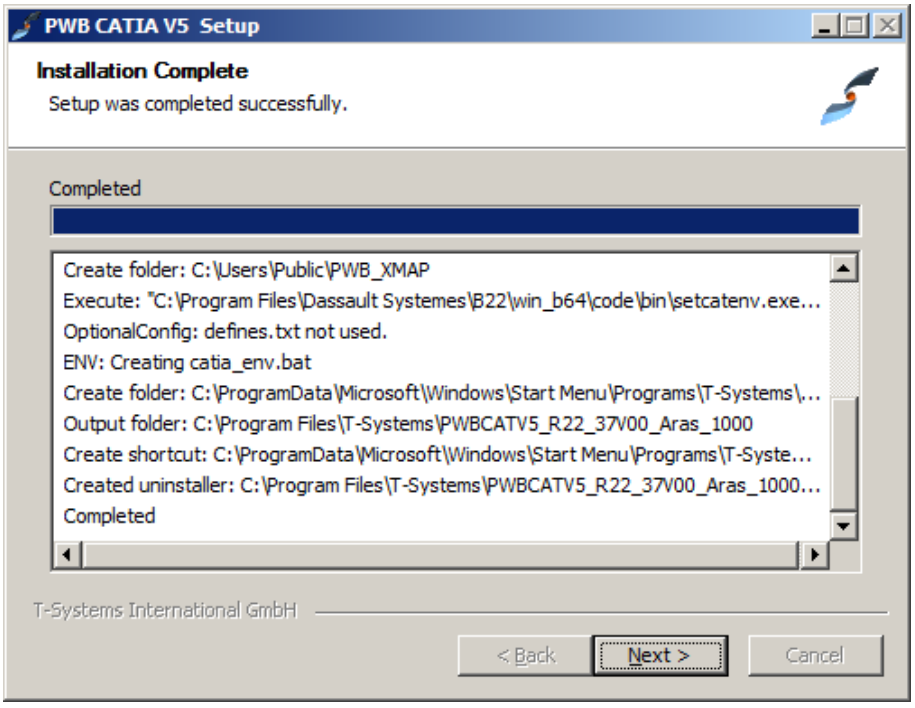


Picture 11: Location of SOAP Target URL

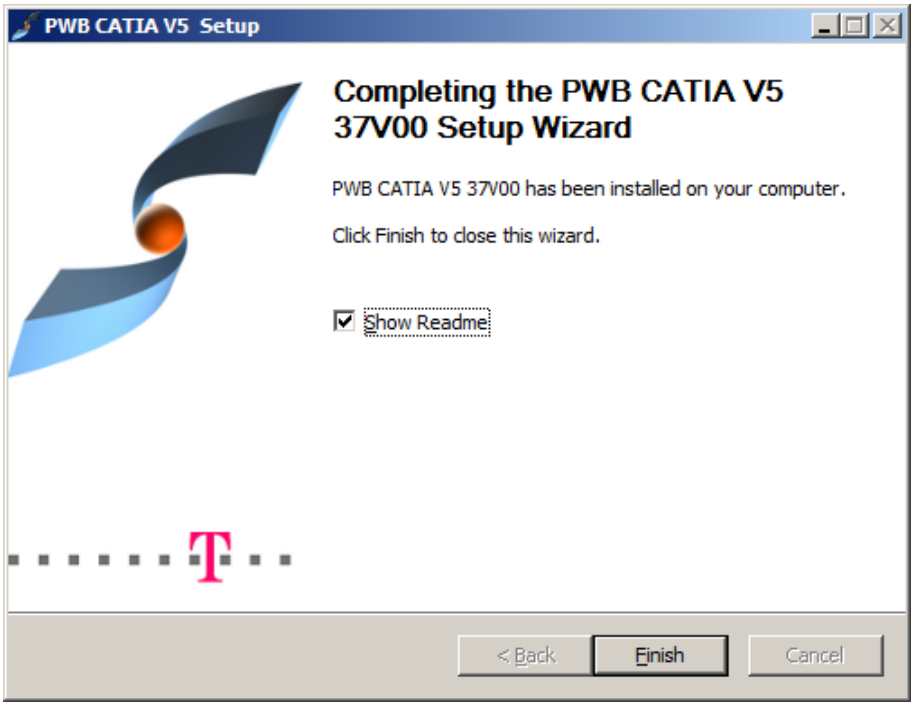
The installer will proceed in its process. The taken actions will be journalized (see *Picture 12: Installation progress* and *Picture 13: Installation progress end*).



Picture 12: Installation progress



Picture 13: Installation progress end


PWB CATIA V5 Setup
Completing the PWB CATIA V5 37V00 Setup Wizard
PWB CATIA V5 37V00 has been installed on your computer.
Click Finish to close this wizard.
Show Readme
< Back
Finish
Cancel

Picture 14: Installation finished

Silent Installation

It is possible to use a silent installation for the client installation.

Parameters

The following parameters are available for the silent installation:

Parameter name	Description	Sample value
/S	Activates the silent mode.	
/User= value	Installation only for yourself ("User") or for all users of the computer ("Admin"). Default is the highest possible value.	Admin
/PdmPackageNamePath = (File full path)	The full path of the zip file which includes the PDM package.	D:\PWBCATV5_37V00_Aras_1000.zip
/D= (Directory path)	The target directory of the installation.	C:\Program Files\T-Systems\PWBCATV5_R22_37V00_Aras_1000
/CatiaInstDir= (Directory path)	The directory of the CATIA V5 installation.	C:\Program Files\Dassault Systemes\B22
/CatiaEnvFile= (File full path)	The full path of the currently used CATIA environment file.	C:\Documents and Settings\All Users\Application Data\DassaultSystemes\CATEnv64\CATIA.V5-

		6R2012_CATIAV5R2264.B22.txt
/ExchangeMap= (Directory path)	The directory of the Exchange Directory.	C:\Users\Public\PWB_XMAP
/SoapTargetURL = (URL)	The SOAP target URL of the Aras server.	http://localhost:80/InnovatorServer

The parameters “/S” and “/SoapTargetURL” are required.

The parameter “/User” is optional. The highest possible value will be used as default value.

The parameter “/PdmPackageNamePath” is optional if the installation had run before. Then the last package will be used. If it is the first installation you have to provide the file path of the PDM package.

The parameter “/D” is optional. A part of the value will be taken from the current directory.

The both values for the CATIA installation are optional; the values can be fetched from the Windows registry.

The parameter “/ExchangeMap” is optional. The directory “C:\Users\Public\PWB_XMAP” will be used as default value.

If one value is not given and it is not possible to fetch a value from the system the installation process will be stopped and the error message can be found in the file *install.log*.

Usage

For the silent installation please open a command line window as administrator.

Inside the temporary installation location, locate the folder “PWBCATV5_[Rxx]_[xxVxx]\install\windows” for an installation on a client with Windows 32 Bit or the folder “PWBCATV5_[Rxx]_[xxVxx]\install\windows _64” for an installation on a client with Windows 64 Bit.

Start the silent installation with a command line like this example:

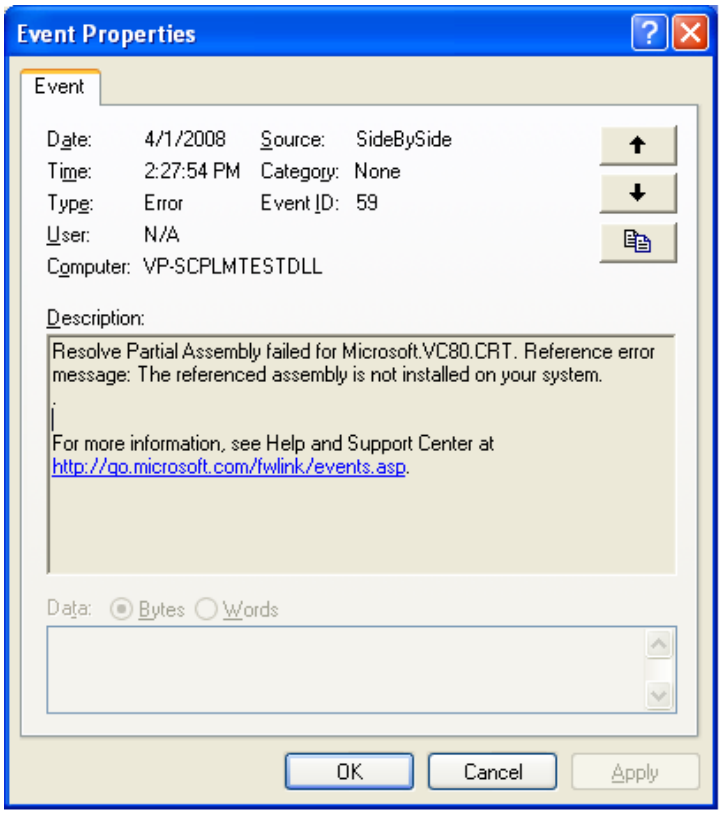
```
Setup.exe /S /User= Admin /PdmPackageNamePath=
D:\PWBCATV5_37V00_Aras_1000.zip /D= "C:\Program Files\T-Systems\
PWBCATV5_R22_37V00_Aras_1000" /CatiaInstDir= "C:\Program
Files\Dassault Systemes\B22" /CatiaEnvFile= "C:\Documents and
Settings\All Users\Application Data\DassaultSystemes\CATEnv64\
CATIA.V5-6R2012_CATIAV5R2264.B22.txt" /ExchangeMap=
"C:\Users\Public\PWB_XMAP" /SoapTargetURL=
"http://localhost:80/InnovatorServer"
```

The log file of the installation will be stored in the current directory. There you can find the information about the installation process.

When the installation ended successful you will find the success message in this file.

Troubleshooting R18 (R17 (64 Bit))

If your PDM Workbench Toolbar does not appear when starting up CATIA V5, please check Windows Event Viewer → System. Check if there is a SideBySide error like this.


Event Properties
Event
Date: 4/1/2008 Source: SideBySide
Time: 2:27:54 PM Category: None
Type: Error Event ID: 59
User: N/A
Computer: VP-SCPLMTESTDLL
Description:
Resolve Partial Assembly failed for Microsoft.VC80.CRT. Reference error message: The referenced assembly is not installed on your system.
For more information, see Help and Support Center at http://go.microsoft.com/fwlink/events.asp.
Data: Bytes Words
OK
Cancel
Apply

Picture 15: Event Properties

This error indicates that you need additional Windows runtime libraries.

You can find these runtimes in one of the following directories:

```
PWBCATV5_R17_32V00\Windows_Runtime\x64\
(required for 64 Bit CATIA only)
PWBCATV5_R18_32V00\Windows_Runtime\x64\
(64 Bit CATIA)
PWBCATV5_R18_32V00\Windows_Runtime\x86\
(32 Bit CATIA)
```

There are two possibilities to install the new runtime:

Install the runtime libraries into the Windows installation (recommended).

This may need system privileges.

For 32 Bit PWB/CATIA you have to extract the package `vcredist_x86.zip` and execute the setup routine `vcredist_x86\vcredist_x86.exe`

For 64 Bit PWB/CATIA you have to extract the package `vcredist_x64.zip` and execute the setup routine `vcredist_x64\vcredist_x64.exe`

or

Copy the additional libraries in a sub directory of the PWB CATIA V5 installation.

For 32 Bit PWB/CATIA you have to extract the package `Microsoft.VC80_x86.zip`. Copy the folder `Microsoft.VC80.CRT` to the binary location of the PWB CATIA module: `PWBCATV5_R18_32V00\intel_a\code\bin\Microsoft.VC80.CRT`

For 64 Bit PWB/CATIA you have to extract the package `Microsoft.VC80_x64.zip`. Copy the folder `Microsoft.VC80.CRT` to the binary location of the PWB CATIA module: `PWBCATV5_R18_32V00\win_b64\code\bin\Microsoft.VC80.CRT`

Testing the installation

Windows

Use: **Start→Programs→T-Systems→PWBCATV5_R22_37V00→PWB_START** to launch CATIA V5.

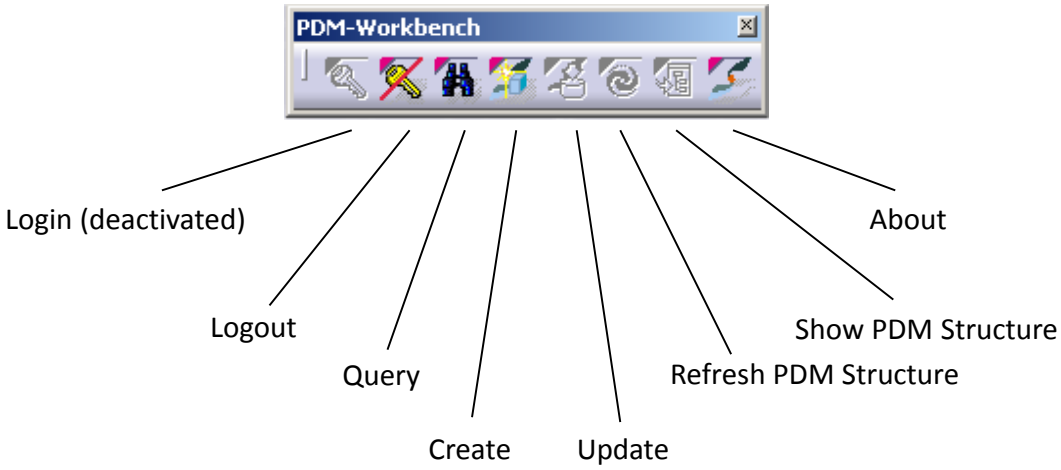
After the CATIA V5 has started the following message should appear in the command window:

```
debug on (level 1)
PDM Workbench: Module Number 1030 :
license successfully allocated
```

The License Module Number may vary.

Within CATIA V5 the following toolbar has to be visible:

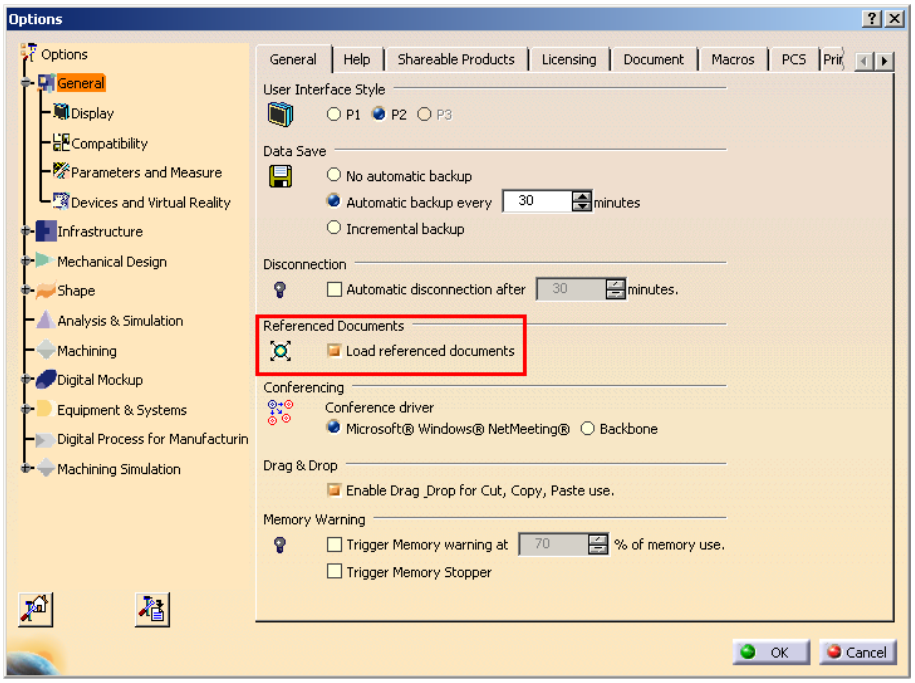
Picture 16: PDM Workbench toolbar before the login



Picture 17: The PDM Workbench toolbar after the login

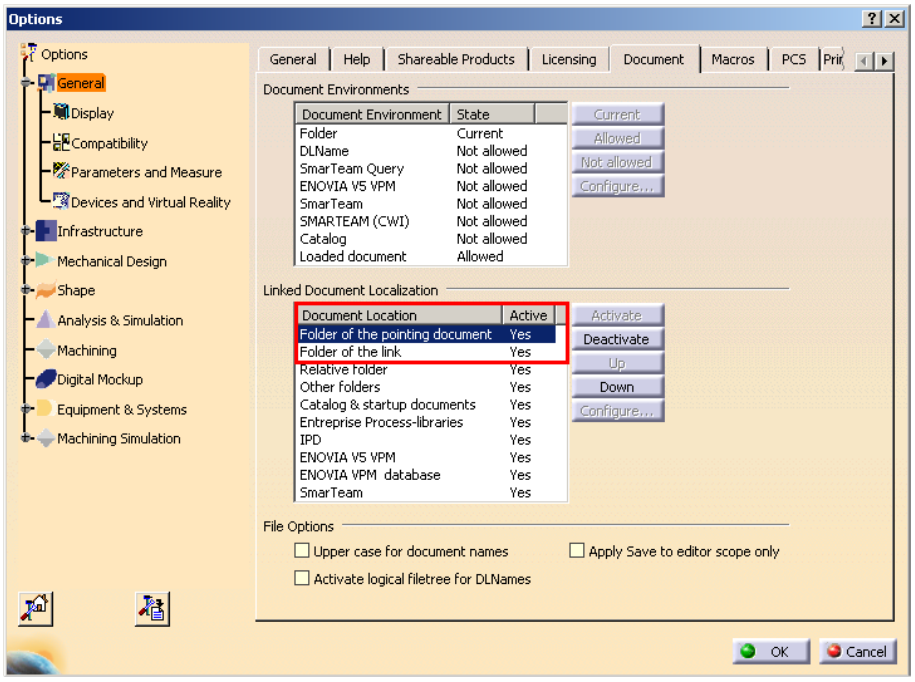
In the CATIA V5 Settings the following options have to be set as described below:

The *Load Referenced documents* option must be set in **Tools→Options→General** settings (see *Picture 18: CATIA V5 General→General Settings*).



Picture 18: CATIA V5 General→General Settings

In the *Linked Document Localization* the Options ***Folder of the pointing document*** and ***Folder of the link*** must be set to yes, and should be in this order (see *Picture 19: CATIA V5 General→Document Settings*).



Picture 19: CATIA V5 General→Document Settings

For PDM Workbench functionality please refer to the *PDM Workbench User Manual.*

Switching to the PWBSchema file suited for BOM Part Structures

The PDM Workbench installation package for the Aras Innovator clients also contains the PWB schema file suited to the BOM Part Structure data model. It is named “PWBSchema_Aras_PartStruc” and it resides in the ‘config’ directory, beside the files “PWBSchema.dtd” and “PWBSchema.xml”.

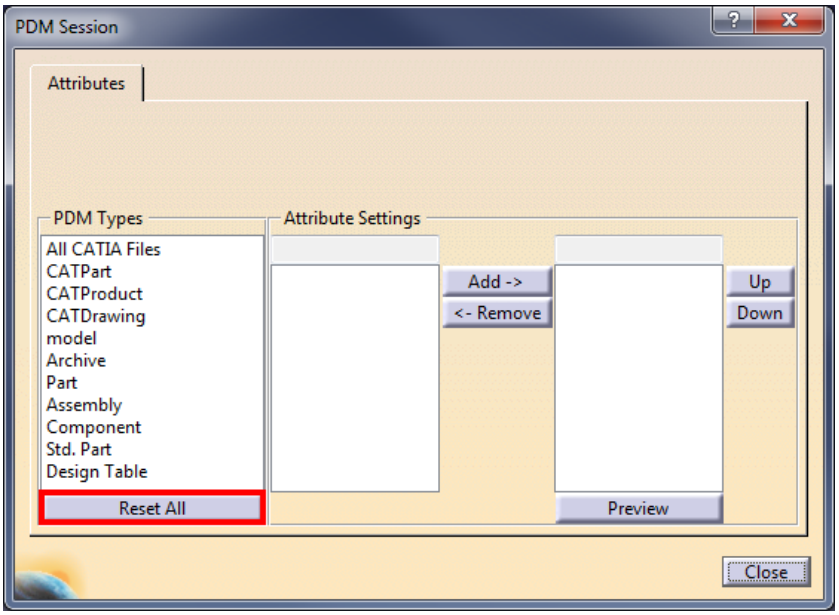
In order to use the BOM Part Structure data model please replace the file “PWBSchema.xml” in the installed PDM Workbench client directory (e.g. “C:\Program Files\T-Systems\ PWBCATV5_R22_37V00_Aras_1000\config\PWBSchema.xml”) with the other configuration file. Please also make sure that the PDM Workbench startup file “pwb_start.bat” contains the correct reference to the schema file (e.g. “SET PWB_SCHEMA_FILE=%PWBDIR%\config\PWBSchema.xml”).

PWBSchema modification

In case of an update of the CATIA client or changes in the PDM Workbench configuration file PWBSchema.xml it is necessary to refresh the List View Column definition for all classes.

Please log in into the PDM system.

Choose in CATIA V5 **Tools→Options** and there *General – Compatibility – PDM Workbench*. Click on “Customize List View”. The Configuration dialog opens (see *Picture 20: PDM Session Configuration dialog*).



Picture 20: PDM Session Configuration dialog

Please click the button “Reset All” in order to refresh the changes from the configuration file.

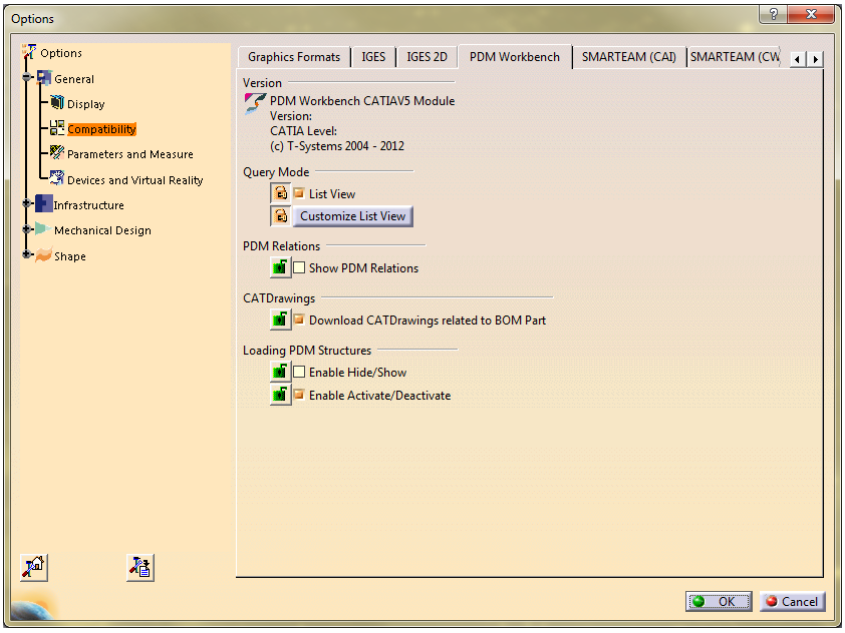
Setting of Environment Variables

The PDM Workbench software will use the following environment variables in the CATIA environment:

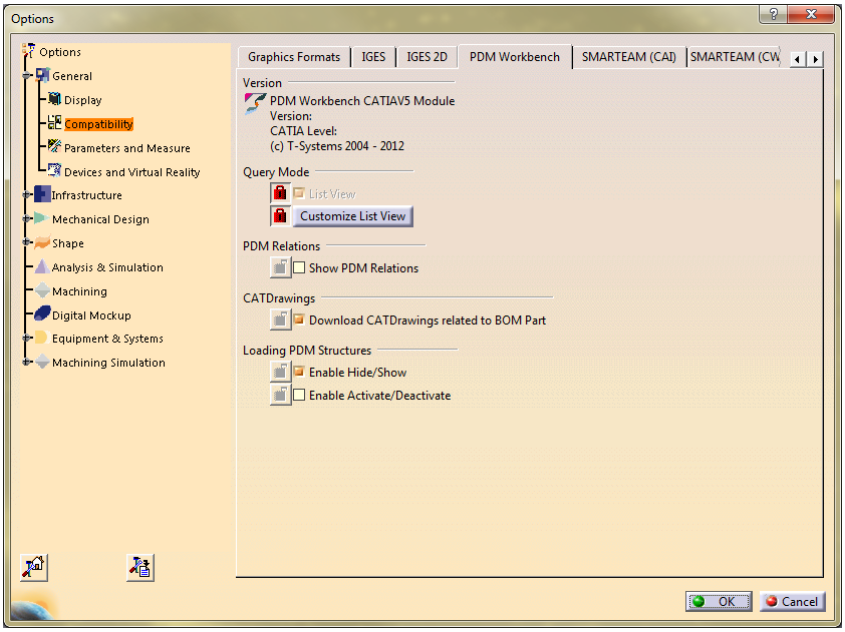
Environment variable	Comment
PWB_XMAP	The location of the exchange map directory. The exchange map directory must be unique for every started CATIA session on the same client.
PWB_SCHEMA_FILE	Path including file name of the XML configuration file.
PWB_SOAP_TARGET_URL	The URL of the web service. Host and Port, e.g. edmg119:9070
PWB_DEBUG	Set to "ON" to receive PWB debug output in the console.
PWB_ADDTEMP_PREFIX	The prefix for the rename of the Part Numbers and File Names for the "Add Temp" and "Open File Temporary" command. Default value is "TMP".

Administrative lock for PDM Workbench Preferences

An administrator can lock the PDM Workbench preferences similar to other CATIA preferences, using the CNEXT –admin interface:



Picture 21: PDM Workbench preferences (administrator view)



Picture 22: PDM Workbench preferences (user view)

The locked preferences will have the administrator's default values and cannot be changed by the users.

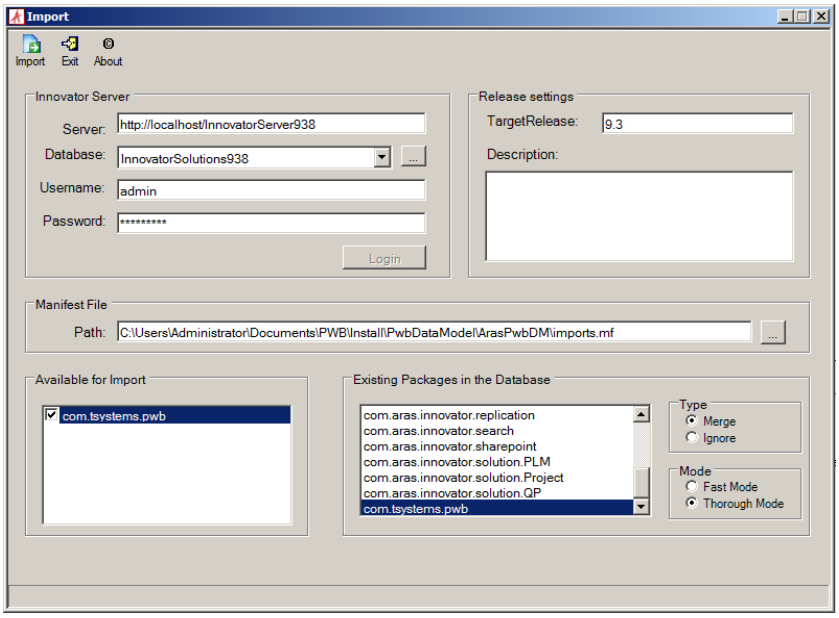
CHAPTER 3

PDM Workbench Data Model

Installation

The PDM Workbench data model and several server-side methods which call and support the main server functionality defined in the PDM Workbench server DLL (see chapter 4) need to be imported to Aras Innovator.

For this the “PwbDataModel.zip” file needs to be unpacked first. Then three packages need to be imported to Aras Innovator with the Aras Innovator import utility[1]:



Picture 23: PDM Aras Innovator import utility

Please select the manifest files

- PwbDataModel\ArasPwbDM\imports.mf
- PwbDataModel\ArasPwbDM_PLM\imports.mf
- PwbDataModel\ArasPwbDM_Core\imports.mf

in the import utility in this order and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 23: PDM Aras Innovator import utility*).

[1] The import utility has to be downloaded from the Aras homepage and to be installed. Link: www.aras.com → Community → Projects → Package Import Export Utility

CHAPTER 4

PDM Workbench Server DLL

Copying the DLL

Please copy the files

```
PwbServerAddin.dll
PwbServerAddin.pdb (optional)
```

from the distribution package to the Aras Innovator server directory

```
C:\Program Files\Aras\Innovator\Innovator\Server\bin
```

or to the corresponding directory if the Aras Innovator server has been installed in a different directory.

Modifying the server configuration file

Also, please modify the file

`C:\Program Files\Aras\Innovator\Innovator\Server\method-config.xml`

by adding the highlighted lines:

```
...
<MethodConfig>
  <ReferencedAssemblies>
    <name>System.dll</name>
    <name>System.XML.dll</name>
    <name>System.Web.dll</name>
    <name>System.Data.dll</name>
    <name>$(binpath)/IOM.dll</name>
    <name>$(binpath)/InnovatorCore.dll</name>
    <name>$(binpath)/CoreCS.dll</name>
    <name>$(binpath)/SPConnector.dll</name>
    <name>$(binpath)/PwbServerAddin.dll</name>
  </ReferencedAssemblies>
...

...
<Template name="CSharp" line_number_offset="39"><![CDATA[
using System;
using System.IO;
using System.Xml;
using System.Text;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Net;
using System.Web;
using System.Web.SessionState;
using System.Globalization;
using Aras.IOM;
using PwbServerAddin;
using PwbServerAddin.Base;

namespace $(pkgname)
{
...
```

CHAPTER 5

Client Customization

The display names in the CATIA V5 workshop can be changed.

The data model of the backend PDM system has to be defined for the CATIA V5 workshop.

The variable $CATIA_INSTALL_DIR defines the installation directory of the PDM Workbench CATIA client.

Display names

The Native Language Support (NLS) files for the CATIA V5 workshop are placed in the following directory:

```
$CATIA_INSTALL_DIR\intel_a\resources\msgcatalog
```

or

```
$CATIA_INSTALL_DIR\win_b64\resources\msgcatalog
```

and the sub directories for the different languages.

There are several NLS files for the dialogs and commands.

The displays for the PDM schema file (see above) are defined in the following files:

- PWBSchemaDisplayNames.CATNls
- PWBSchemaDisplayNames_Aras_Aras.CATNls where the first “Aras” corresponds with the “system” value and the second "Aras" corresponds with the “customization” value of the PDM systems in the PDM schema file.

The displays for the error messages are defined in the following file:

- PWBUserErrors.CATNls

The displays in these NLS files can be changed.

Icons

The icons for the objects for the CATIA V5 workshop are placed in the following directory:

```
$CATIA_INSTALL_DIR\intel_a\resources\graphic\icons\normal
```

or

```
$CATIA_INSTALL_DIR\win_b64\resources\graphic\icons\normal
```

The icon names correspond to the names used in the PDM schema file.

Data model definition

The configuration of the data model for Aras Innovator has to be done in the configuration file (xml schema file) to be used by the PDM Workbench module within CATIA V5.

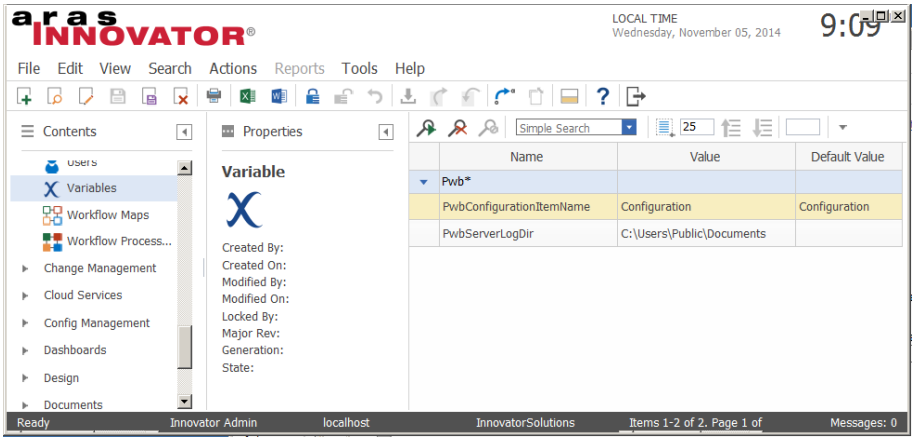
CHAPTER 6

Server Configuration

This chapter describes the configuration of the server side of the PDM Workbench integration.

Configuration Variables

The following Aras Innovator server configuration variables need to be set for PDM Workbench to work correctly:



Picture 24: Aras Innovator server configuration variables

- PwbConfigurationItemName

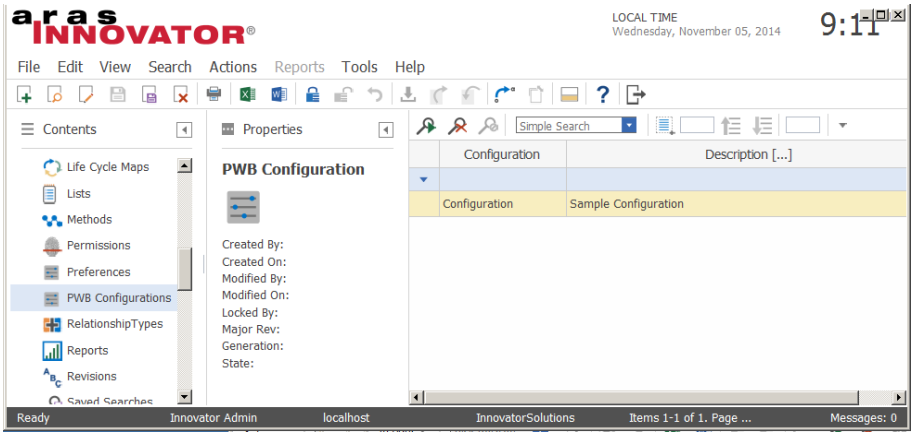
  The name of the PDM Workbench configuration item which contains additional configuration information, like the attribute mapping configuration. Please see “Configuration Items” for more details.

- PwbServerLogDir

  The absolute path of the directory into which the server log file should be written. If this variable is empty then no server log file will be written.

Configuration Items

In order to define the environment variables and to configure the mapping of attributes between Aras Innovator and CATIA V5 a special configuration item (see *Picture 25: PWB Configuration item in Aras Innovator*) has to be used:



Picture 25: PWB Configuration item in Aras Innovator

CHAPTER 7

Configurations for specific functionalities

This chapter describes the configuration of the PDM Workbench for specific functionalities.

Standard configuration

Exchange map

In the Schema file the absolute path of the exchange map directory, where the downloaded CATIA files are stored, can be configured.

Example:

```
<xmap value="C:\PWB_XMAP" />
```

If the exchange map value is defined by the environment variable "PWB_XMAP", then that takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

Optional.

SOAP target URL

In the Schema file the URL of the server process, that the PDM Workbench client uses for its SOAP requests, can be configured.

Example:

```
<soapTargetUrl value="http://hostname:80/InnovatorServer" />
```

If the soap target URL value is defined by the environment variable "PWB_SOAP_TARGET_URL", then that takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

Optional.

SOAP client call path environment variable

In the Schema file the name of the environment variable which contains the path to the Aras Innovator SOAP client executable has to be configured.

Example:

```
<externalSoapClientCallPathEnvVar
   value="PWB_SOAPCLIENT_PATH_ARAS" />
```

This variable is defined in the start script of the PDM Workbench, e.g.

```
SET PWB_SOAPCLIENT_PATH_ARAS=
%PWBV5DIR%\code\bin\PwbArasSoapClient.exe
```

Mandatory.

File client call path environment variable

In the Schema file the name of the environment variable which contains the path to the Aras Innovator file client executable has to be configured.

Example:

```
<externalFileClientCallPathEnvVar
   value="PWB_FILECLIENT_PATH_ARAS" />
```

This variable is defined in the start script of the PDM Workbench, e.g.

```
SET PWB_FILECLIENT_PATH_ARAS=
%PWBV5DIR%\code\bin\PwbArasFileClient.exe
```

Mandatory.

PWB Window color

In the Schema file the background color of the PWB window can be configured. This tag contains the red, green, and blue values (0 - 255) of the color.

Example:

```
<pwbWindowColor red="143" green="155" blue="177" />
```

If not set then the CATIA V5 standard color will be used.

Optional.

Maximum expansion level

In the Schema file the maximum expansion level can be defined. It describes how many levels, starting from the root node, will be expanded by the PDM Workbench.

Example:

```
<maxExpansionLevel value="30" />
```

If not set then all levels of the assembly will be expanded.

Optional.

Read only

In the Schema file it can be configured that the loaded CATIA files are set to read-only if the PDM nodes are not modifiable.

Example:

```
<setReadOnly value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Reload after update

In the Schema file it can be configured that the CATIA document will be reloaded after the update.

Example:

```
<neverReloadCatiaDocAfterUpdate value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Check file versions in structure

In the Schema file it can be configured that the file versions should be checked in the structure.

Example:

```
<checkFileVersionsInStructure value="true" />
```

Default value: true

Optional. Possible values: “true”, or “false”.

Check CATParts in Visualization mode

In the Schema file it can be configured that it will be checked for CATParts loaded in Visualisation mode..

Example:

```
<checkCatPartsInVisuMode value="false" />
```

Default value: true

Optional. Possible values: “true”, or “false”.

Check CATIA PartNumbers before update

In the Schema file it can be configured that the CATIA part numbers are checked for invalid characters before update.

Example:

```
<checkCatiaPartNumbersBeforeUpdate value="false" />
```

Default value: true

Optional. Possible values: “true”, or “false”.

Allow loading different file versions

In the Schema file it can be configured that it is allowed to load different file versions in the CATIA session.

Example:

```
<allowLoadingDifferentFileVersion value="false" />
```

Default value: true

Optional. Possible values: “true”, or “false”.

Show out-of-date link info

In the Schema file it can be configured that the relation should be marked as out-dated, if a newer object exists.

Example:

```
<showOutOfDateLinkInfo value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Update after create new CATIA file

In the Schema file it can be configured that the updated should be started after the create of a new CATIA file.

Example:

```
<startUpdateAfterCreateNewCatiaFile value="false" />
```

Default value: true

Optional. Possible values: “true”, or “false”.

Upload changed files at sync failure

In the Schema file it can be configured that the changed files are uploaded even if an error occurs in the update process.

Example:

```
<uploadChangedFilesAtSyncFailure value="false" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Upload uses Check in Manager

In the Schema file it can be configured that the check in manager will be used.

Example:

```
<uploadUsingCheckinManager value="false" />
```

Default value: true

Optional. Possible values: “true”, or “false”.

Required attributes

In the Schema file it can be configured if the wildcard (e.g. “*”) is an invalid value for required attributes.

If this setting is active it defines values as not valid for required query dialog attributes. For instance, if the item_number attribute is defined as required, then the setting above defines the query value of “*” as invalid for that attribute.

Example:

```
<requiredDialogAttributes value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Maximum description attribute length

In the Schema file the maximum length of the description attribute can be defined.

Example:

```
<maximumDescriptionAttributeLength value="10" />
```

Optional.

Show data source NLS values

In the Schema file it can be configured if the NLS values of the data source should be shown.

Example:

```
<showDataSourceNlsValues value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Rename existing files in exchange map

In the Schema file it can be configured if the existing files in the exchange map should be renamed at load.

Example:

```
<renameExistingExMapFilesAtLoad value="false" />
```

Default value: true

Optional. Possible values: “true”, or “false”.

Auto login on session expired

In the Schema file it can be configured if an auto login should be performed if the session has been expired..

Example:

```
<autoLoginOnSessionExpired value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Session settings

In the Schema file the session settings of the PDM Workbench can be defined. The following entries are supported.

Example:

```
<sessionSettings>
   <-- Set one of these two values as default.
       It can be changed in the CATIA session with
       Tools->Options->Compatibility->PDM Workbench. -->
   <queryMode name="listViewWindow" />
   <queryMode name="pwbWindow" />
   <-- Set one of these two values as default.
       It can be changed in the CATIA session with
       Tools->Options->Compatibility->PDM Workbench. -->
   <relationDisplayMode name="relDisplay" />
   <relationDisplayMode name="noRelDisplay" />
   <-- Set one of these three values. -->
   <passwordEncryption name="WinAuth" />
   <passwordEncryption name="MD5" />
   <passwordEncryption name="none" />
</sessionSettings>
```

Optional.

Language settings

In the Schema file the name and date format of the installed languages (NLS files) have to be configured.

Example:

```
<installedLanguages visibleLength="15">
   <language name="en_us" displayName="NLS_EN"
             dateFormat="PWB_Standard" />
</installedLanguages>
```

Mandatory.

Date format

In the Schema file the definition of the date format for the installed languages has to be stored.

Example:

```
<!-- PWB_Standard = YYYY-MM-DD -->
<dateFormat name="PWB_Standard" separator="-">
   <dateValue name="year" length="4" />
   <dateValue name="month" length="2" />
   <dateValue name="day" length="2" />
</dateFormat>
```

The order of the “dateValue” tags defines the format of the date. For each part of the date the length is defined in the tag.

Mandatory.

Key attribute

Internal attribute, do not change.

Class attribute

Internal attribute, do not change.

Relation attribute

Internal attribute, do not change.

Relationship attribute

Internal attribute, do not change.

Left relationship attribute

Internal attribute, do not change.

Right relationship attribute

Internal attribute, do not change.

Left relation class attribute

Internal attribute, do not change.

Right relation class attribute

Internal attribute, do not change.

Extended relation class attribute

Internal attribute, do not change.

Last modification date attribute

In the Schema file the name of the last modification date attribute can be defined.

Example:

```
<lastModificationDateAttribute name="last_mod_date" />
```

Default value: last_mod_date

Optional.

Mark superseded nodes

In the Schema file it can be configured if superseded nodes should be marked as outdated.

If this setting is set to “true” then superseded nodes (nodes where a newer generation exists) are colored magenta, like outdated nodes, where the CAD file has been modified by another user, instead of black.

Example:

```
<colorSupersededNodesAsOutdated value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Check for CAD owner

If this setting is set to “true” in the Schema file then it will not be possible to load structures which have been created by a different CAD integration (‘external_owner’ attribute value is not ‘T-Systems.Mechanical.CATIA’) .

```
<checkForCadOwner value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Part classes

In the Schema file a list of class names (possibly of size 1) of all part classes can be defined. It has to be defined if the class can have sub parts.

Example:

```
<partClasses>
   <partClassName name="/Part/Assembly" canHaveSubParts="true" />
   <partClassName name="/Part/Component" canHaveSubParts="false"/>
</partClasses>
```

Optional.

Bounding box definition

In the Schema file the names of the bounding box attributes can be defined.

Example:

```
<boundingBoxAttributes>
   <boundingBoxXMinAttr name="x_min" />
   <boundingBoxXMaxAttr name="x_max" />
   <boundingBoxYMinAttr name="y_min" />
   <boundingBoxYMaxAttr name="y_max" />
   <boundingBoxZMinAttr name="z_min" />
   <boundingBoxZMaxAttr name="z_max" />
</boundingBoxAttributes>
```

When all bounding box attributes are defined, the bounding box functionality is enabled, otherwise the functionality is disabled.

Optional.

Toolbar configuration

In the Schema file the toolbar can be configured. It can be defined which toolbar entries have to be removed.

Example:

```
<removeToolbarIcons>
   <icon name="Register" />
   <icon name="Synchronize" />
   <icon name="SetSessionConfig" />
   <icon name="NewPwbWindow" />
   <icon name="DynamicActions" />
</removeToolbarIcons>
```

Optional.

Customer-Specific Environment

If customers use different CATIA releases, they can configure in the start script of the PDM Workbench which environment should be used. The designer will not be able to modify CAD files which have been created in different customer environments.

The CATIA environment can be defined in the Schema file. This definition can be done by one or more attributes, e.g. customer and project.

Example:

```
<catiaEnvironment>
   <envAttribute displayName="NLS_env_customer" pdm="pwb_customer"
                 env="PWB_ENVIRONMENT_CUSTOMER" />
   <envAttribute displayName="NLS_env_project" pdm="pwb_project"
                 env="PWB_ENVIRONMENT_PROJECT" />
</catiaEnvironment>
```

Optional.

The values of these environment variables `PWB_ENVIRONMENT_CUSTOMER` and `PWB_ENVIRONMENT_PROJECT` have to be defined in the start script of the PDM Workbench.

Use case "Query"

The query dialog for PDM objects contains the specific attributes (e.g. `pwb_customer` and/or `pwb_project`) which contain the customer environment or project attribute string.

These attributes are contained read-only so the user can only query for PDM objects which contain the values defined in the environment variables.

Use case "Update"

Whenever the user creates new PDM items the strings from the environment variables are written into the specific PDM item attributes.

Use case "Load"

In the "Load" or "Open" on PDM items a check is peformed whether the value string of the specific item attributes is the same as the one read from the corresponding environment variables.

Only if the value is the same the files will be loaded in CATIA, otherwise the user will get an error message that the environment does not fit.

Data model configuration

The PWB Configuration item setting "UseBomPartStructure" indicates which data model will be used with the PDM Workbench.

The attribute has to be set to "true" in order to use the "BOM Part Structure data model". Otherwise the "Document data model" will be used.

UseBomPartStructure	true

Picture 26: Sample UseBomPartStructure configuration

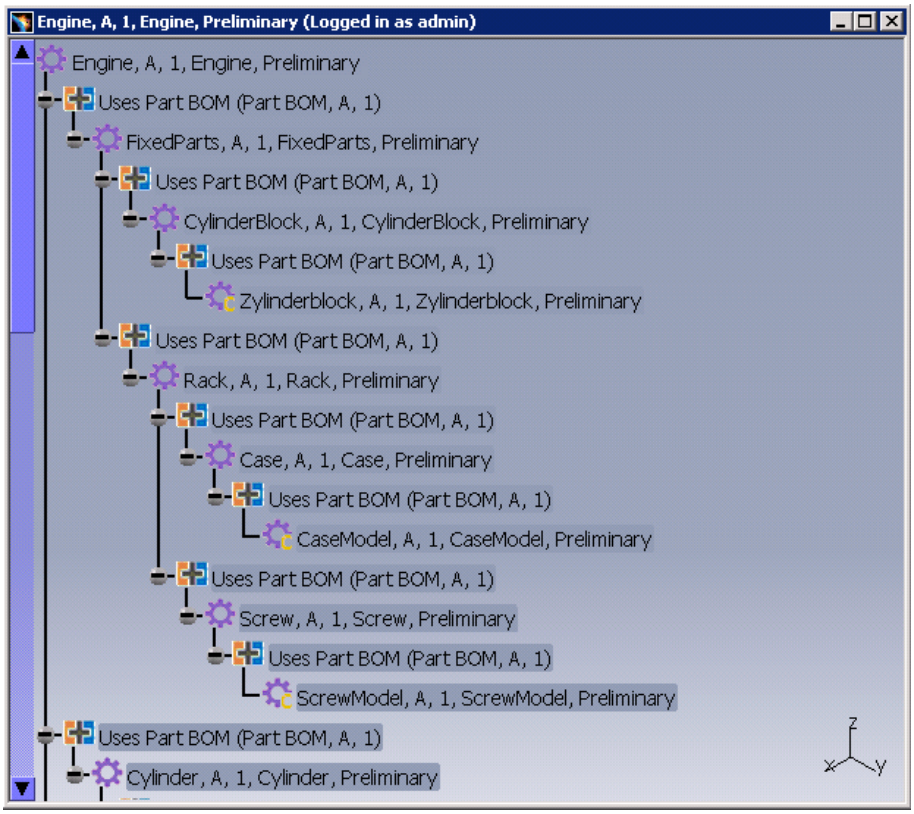
Default value: false

Optional. Possible values: "true", or "false".

BOM Part Structure data model

In the BOM Part Structure data model the PDM structure is represented by Parts (Assembly or Component). The relation "Part BOM" is used.

Each Part is described by a CAD Document which includes the CATIA file for a CATProduct, CATPart, or CATDrawing (see *Picture 27: Structure in the BOM Part Structure data model*).

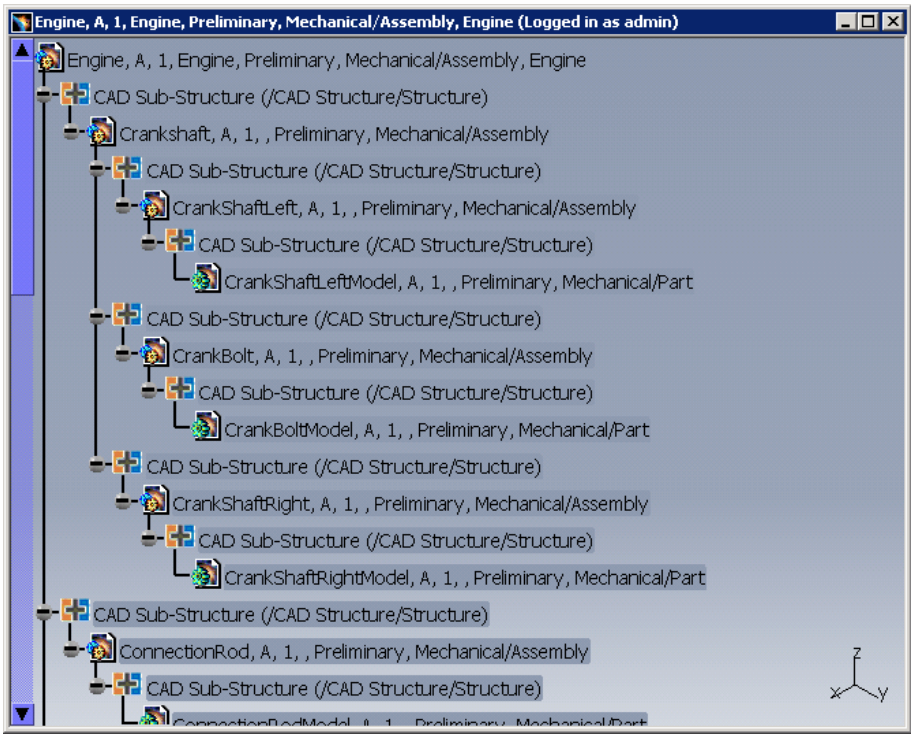

Engine, A, 1, Engine, Preliminary (Logged in as admin)
Engine, A, 1, Engine, Preliminary
Uses Part BOM (Part BOM, A, 1)
FixedParts, A, 1, FixedParts, Preliminary
Uses Part BOM (Part BOM, A, 1)
CylinderBlock, A, 1, CylinderBlock, Preliminary
Uses Part BOM (Part BOM, A, 1)
Zylinderblock, A, 1, Zylinderblock, Preliminary
Uses Part BOM (Part BOM, A, 1)
Rack, A, 1, Rack, Preliminary
Uses Part BOM (Part BOM, A, 1)
Case, A, 1, Case, Preliminary
Uses Part BOM (Part BOM, A, 1)
CaseModel, A, 1, CaseModel, Preliminary
Uses Part BOM (Part BOM, A, 1)
Screw, A, 1, Screw, Preliminary
Uses Part BOM (Part BOM, A, 1)
ScrewModel, A, 1, ScrewModel, Preliminary
Uses Part BOM (Part BOM, A, 1)
Cylinder, A, 1, Cylinder, Preliminary
Z
X
Y

Picture 27: Structure in the BOM Part Structure data model

Document data model

In the Document data model the PDM structure is represented by CAD Documents. The relation "CAD Structure" is used.

Each CAD Document includes the CATIA file for a CATProduct, CATPart, or CATDrawing (see *Picture 28: Structure in the Document data model*).


Engine, A, 1, Engine, Preliminary, Mechanical/Assembly, Engine (Logged in as admin)
Engine, A, 1, Engine, Preliminary, Mechanical/Assembly, Engine
CAD Sub-Structure (/CAD Structure/Structure)
Crankshaft, A, 1, , Preliminary, Mechanical/Assembly
CAD Sub-Structure (/CAD Structure/Structure)
CrankShaftLeft, A, 1, , Preliminary, Mechanical/Assembly
CAD Sub-Structure (/CAD Structure/Structure)
CrankShaftLeftModel, A, 1, , Preliminary, Mechanical/Part
CAD Sub-Structure (/CAD Structure/Structure)
CrankBolt, A, 1, , Preliminary, Mechanical/Assembly
CAD Sub-Structure (/CAD Structure/Structure)
CrankBoltModel, A, 1, , Preliminary, Mechanical/Part
CAD Sub-Structure (/CAD Structure/Structure)
CrankShaftRight, A, 1, , Preliminary, Mechanical/Assembly
CAD Sub-Structure (/CAD Structure/Structure)
CrankShaftRightModel, A, 1, , Preliminary, Mechanical/Part
CAD Sub-Structure (/CAD Structure/Structure)
ConnectionRod, A, 1, , Preliminary, Mechanical/Assembly
CAD Sub-Structure (/CAD Structure/Structure)
Z
X
Y

Picture 28: Structure in the Document data model

BOM Part Structure configuration

The PWB Configuration item setting “BomPartStructureLockUnlockRelatedItem” sets the behavior of “Lock” and “Unlock”.

Normally the “Lock” and “Unlock” context menus in the CATIA window lock and unlock only the CAD document. Also locking and unlocking a part or a CAD document in the PDM structure window only locks the item where the action was performed on. With this configuration setting this behavior changes such that locking a part also locks its related CAD documents, or locking a CATPart or CATProduct CAD document locks its related part, or both of these actions are activated. So, for instance, if this setting is set to “part” or to “both” then in the BOM structure mode locking a CAD document, either explicitly in the PDM structure window, or in the CATIA window, also locks the corresponding part item.

BomPartStructureLockUnlockRelatedItem	cad

Picture 29: Sample BomPartStructureLockUnlockRelatedItem configuration

Default value: both

Optional. Possible values: “part”, “cad”, or “both”.

Query configuration

QueryOrderByAttribute

The PWB Configuration item setting “QueryOrderByAttribute” defines an attribute by which the query results are internally ordered. This is not noticeable by the user, but it can result in significant performance improvements when a query is performed if the attribute is in the database index.

QueryOrderByAttribute	id

Picture 30: Sample QueryOrderByAttribute configuration

Optional.

MaxQueryResults

The PWB Configuration item setting “MaxQueryResults” defines the maximum number of items that are retrieved in a single query. If more items exist the user is informed about that fact.

MaxQueryResults	100

Picture 31: Sample MaxQueryResults configuration

Optional.

Default sort criteria for query results

For every PDM item type an attribute can be configured to be the default sort order attribute. Query results for that item type are automatically sorted by that attribute.

The ‘object’ definition can contain a new optional XML attribute: “defaultSortAttribute”:

```
<object name="Part" displayName="NLS_Part"
        icon="Aras_Part" defaultSortAttribute="item_number" >
```

This attribute has to contain the internal name (not the CATNls name) of an attribute of the item type, e.g. ‘item_number’. The query result will be sorted by the values of that attribute.

Display custom query buttons

In the Schema file the layout of the query dialog can be configured.

If the value is set to “true” the buttons will be display on the left side of the bottom of the dialog. In the default layout (value is “false”) the buttons are on the right side of the dialog.

Example:

```
<displayCustomQueryButtons value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Modal dialog

In the Schema file it can be configured that the query dialog is modal. In this case the dialog has to be closed in order to continue working with the PDM Workbench.

Example:

```
<queryDialogIsModal value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Check if no query attribute filled

In the Schema file it can be configured if a query with empty attributes should be possible.

If the value is set to “false” it is possible the perform a query without filling an attribute in the query dialog.

If the value is set to “true” it is necessary to fill at least one of the attributes in the query dialog.

Example:

```
<checkIfNoQueryAttrsFilled value="false" />
```

Default value: true

Optional. Possible values: “true”, or “false”.

Display string in query list view

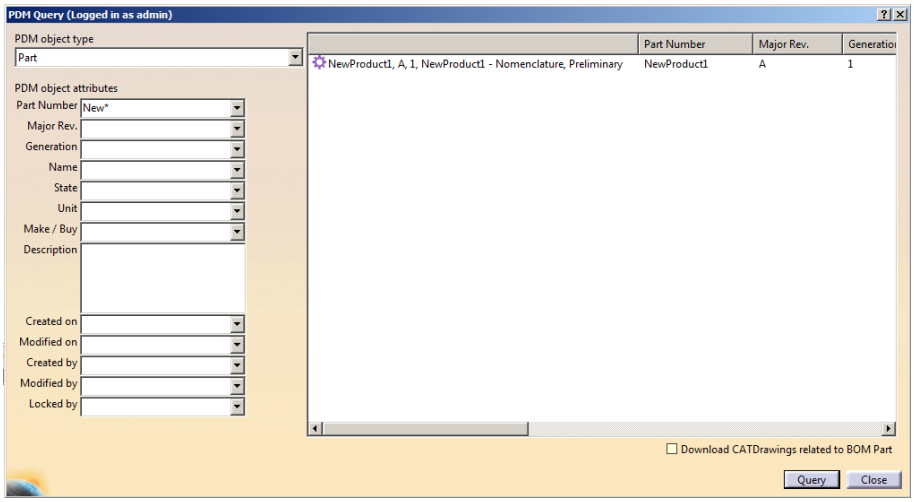
In the Schema file it can be configured if then display string of the object should be shown after the icon in the query list view.

Example:

```
<showDisplayStringInQueryListView value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.



Picture 32: Query list view dialog with Display String

Update configuration

Create dialogs

The PWB Configuration item setting “ShowCreateDialogsDuringUpdate” has to be set to "true" in order to show the create dialogs during the update process.

ShowCreateDialogsDuringUpdate	true

Picture 33: Sample ShowCreateDialogsDuringUpdate configuration

Default value: false

Optional. Possible values: “true”, or “false”.

Create version button

In the Schema file it can be configured if the “Create new file versions at update?” button appears in the update dialog.

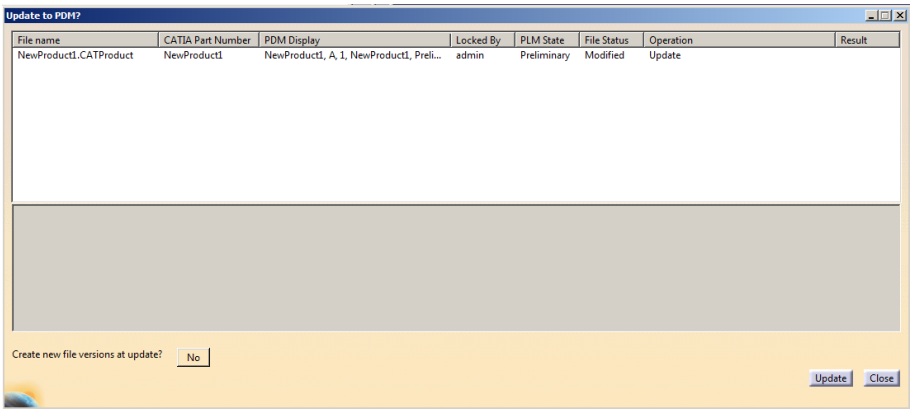
If the value of this setting is set to “false” the button does not appear in the update dialog. The default value (overwriting the existing document) will always be active.

Example:

```
<showCreateVersionAtUpdate value="false" />
```

Default value: true

Optional. Possible values: “true”, or “false”.



Picture 34: Update dialog with "Create new file versions at update?"

Default create version value

In the Schema file the default value of the "Create new file versions at update?" button can be configured.

If the value of this setting is set to "false" the default value of the button is "No".

If the value of this setting is set to "true" the default value of the button is "Yes".

Example:

```
<defaultCreateVersionAtUpdateValue value="true" />
```

Default value: false

Optional. Possible values: "true", or "false".

Show create parts

In the Schema file it can be configured if the "Create new parts at update?" button appears in the update dialog.

If the value of this setting is set to "false" the button does not appear in the update dialog.

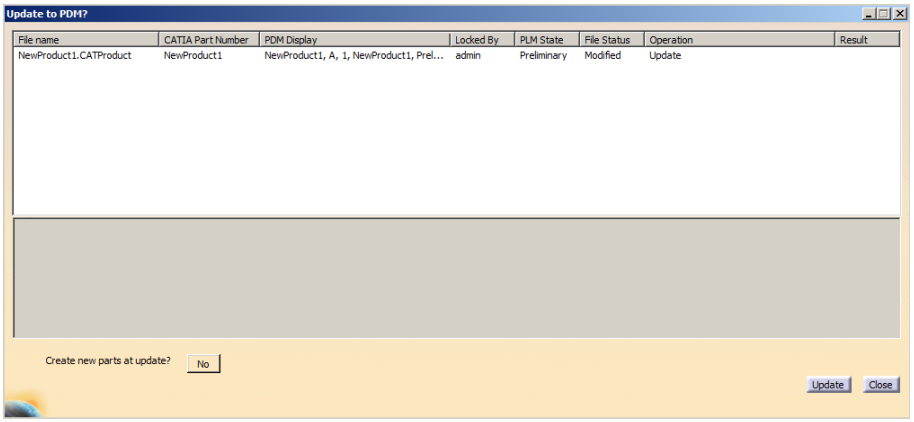
If the value of this setting is set to "true" the button appears in the update dialog.

Example:

```
<showCreatePartsAtUpdate value="true" />
```

Default value: false

Optional. Possible values: "true", or "false".



Picture 35: Update dialog with "Create new parts at update?"

Default create parts value

In the Schema file the default value of the “Create new parts at update?” button can be configured.

If the value of this setting is set to “false” the default value of the button is “No”.

If the value of this setting is set to “true” the default value of the button is “Yes”.

Example:

```
<defaultCreatePartsAtUpdateValue value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Revise configuration

Modifiable attributes

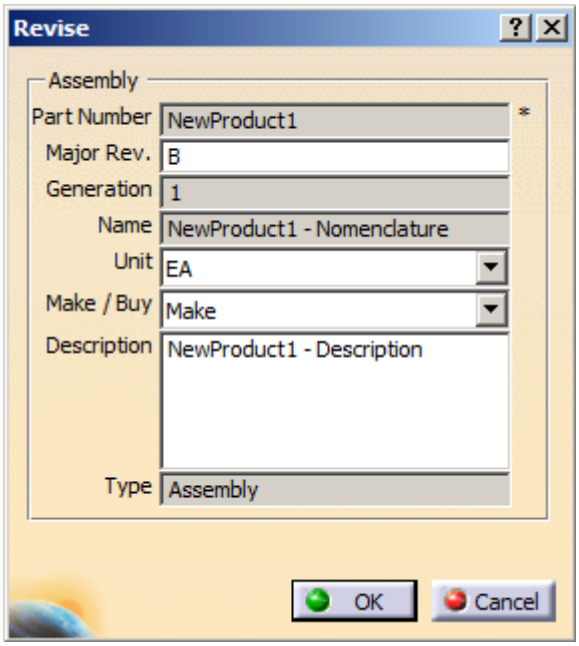
In the Schema file it can be configured which attributes can by additionally modified in the revise dialog.

Example:

```
<attrsModifiableAtRevise value="major_rev|description" />
```

These attribute values can be combined with the pipe "|".

Optional.



Picture 36: Revise dialog with modifiable attributes “major_rev” and “description”

Expand structure

In the Schema file it can be configured if the structure should be expanded before the revise.

Example:

```
<expandStructureForRevise value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Numbering

CadDocNumberAttr

The PWB Configuration item setting “CadDocNumberAttr” contains the CAD document number of the document class.

CadDocNumberAttr	item_number

Picture 37: Sample CadDocNumberAttr configuration

Default value: item_number

Optional.

PartNumberAttr

The PWB Configuration item setting “PartNumberAttr” contains the part number of the part class.

PartNumberAttr	item_number

Picture 38: Sample PartNumberAttr configuration

Default value: item_number

Optional.

Initial version strings

InitialVersionCadDoc

The PWB Configuration item setting “InitialVersionCadDoc” is the initial version string for the item type "CAD Document".

InitialVersionCadDoc	A

Picture 39: Sample InitialVersionCadDoc configuration

Optional.

InitialVersionPart

The PWB Configuration item setting “InitialVersionPart” is the initial version string for the item type " Part ".

InitialVersionPart	A

Picture 40: Sample InitialVersionPart configuration

Optional.

Promote states

PromoteSourceStates

The PWB Configuration item setting “PromoteSourceStates” is a list of the promote source states, separated by “|”.

PromoteSourceStates	Preliminary|In Review

Picture 41: Sample PromoteSourceStates configuration

Default value: Preliminary|In Review

Optional.

PromoteTargetStates

The PWB Configuration item setting “PromoteTargetStates” is a list of the promote target states, separated by "|".

PromoteTargetStates	In Review|Released

Picture 42: Sample PromoteTargetStates configuration

Default value: In Review|Released

Optional.

Standard Part Functionality

StandardPartAdmin

The PWB Configuration item setting “StandardPartAdmin” defines the standard part administrator identity.

Mandatory. Can be removed if the following server events are removed:

PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts
PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts
PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts
PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts
PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts

Standard Part functionality for BOM Part Structure data model

In part structure mode, it is possible to define part items and their corresponding CAD document items as standard parts. Standard parts are supposed to be parts which are used in a wide variety of different contexts and which are generally not modified by the designer, only used in the product structures that the designer works on.

Only a standard part administrator can create or modify standard parts. A standard part administrator is defined as every identity which belongs to the identity which is set as the setting “StandardPartAdmin”. By default it is the identity “Standard Part Administrator”, which is added with the PDM Workbench installation:

StandardPartAdmin	Standard Part Administrator

Picture 43: Sample StandardPartAdmin configuration

By default the “Innovator Admin” identity is a standard part administrator.

The CATParts which are related to standard parts can either be downloaded like the CATIA files which are related to other parts, or they can reside in a local directory which is accessible to CATIA V5.

This setting in the PWBSchema.xml file defines that the standard part CATParts are downloaded to the PWB_XMAP directory:

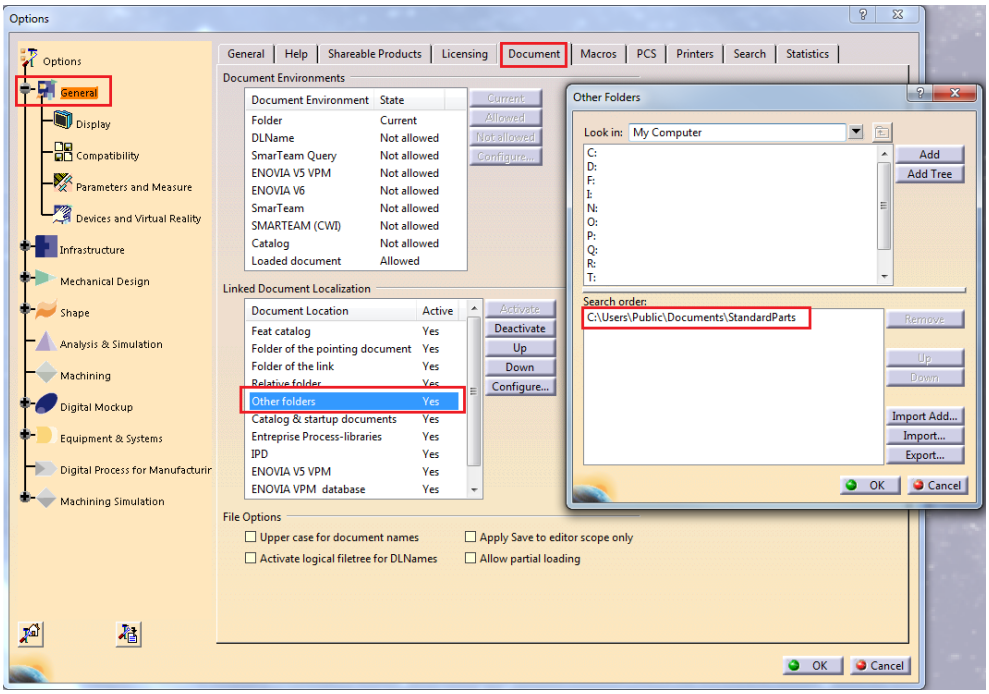
```
<standardPartFiles download="true" />
```

This setting defines that they are not downloaded, but taken from a local directory:

```
<standardPartFiles download="false"
                   localDir="C:\Users\Public\Documents\StandardParts" />
```

For this setting three conditions need to be fulfilled:

1. The defined directory is accessible in the CATIA V5 environment.
2. The standard part CATPart files reside in that directory.
3. CATIA V5 needs to be set up to be able to use files from that directory.



Picture 44: Making CATParts in a local folder accessible

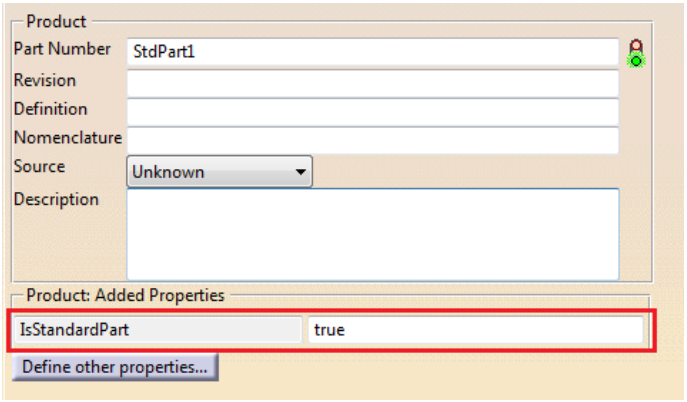
Another setting makes it possible to import structures containing standard part CATParts to PDM even if those standard parts have not been loaded in the PDM Workbench session.

Normally CATIA files which are not loaded from PDM are treated as new items, and new CAD documents and parts are created for them in the update process. If CATParts contain a certain definition as a user-defined attribute they are treated as standard parts, and the update process queries for existing standard part items in the database instead of trying to create a new part item. If such a standard part item is found it is related to the parent part.

In the PWBSchema.xml file it is possible to define the name of the user-defined attribute which defines a standard part CATPart file, for instance ‘IsStandardPart’:

```
<standardPartUserDefPropAttribute name="IsStandardPart" />
```

A user-defined attribute with the same name has to be defined in the CATPart file. If the value is any of the strings “true”, “TRUE”, or “1”, then the CATPart is considered to be a standard part CATPart.



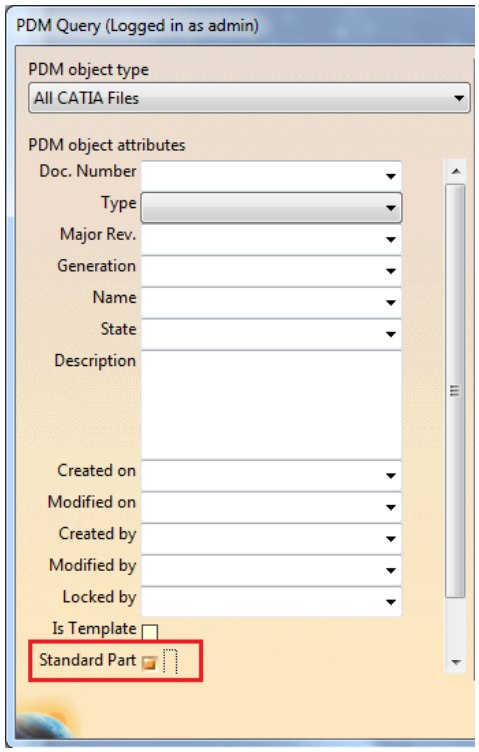
Picture 45: Defining a CATPart as a standard part

In that case the update process does not try to create a new part which corresponds to the CATIA document, but it queries for an existing standard part in PDM instead.

Standard Part functionality for Document data model

The standard part functionality has been extended to work with CAD document structures.

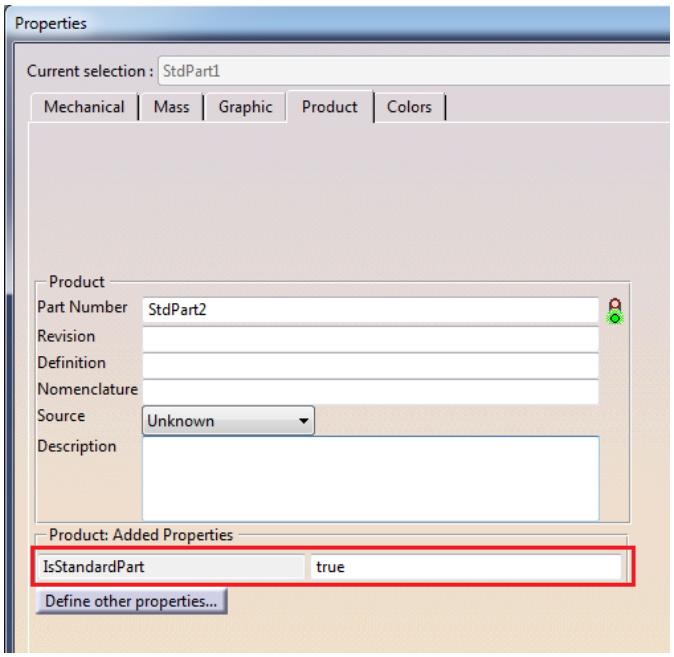
There is a new check box which defines whether a CAD document is defined as a standard part item:



Picture 46: “Standard Part” check box for CAD documents

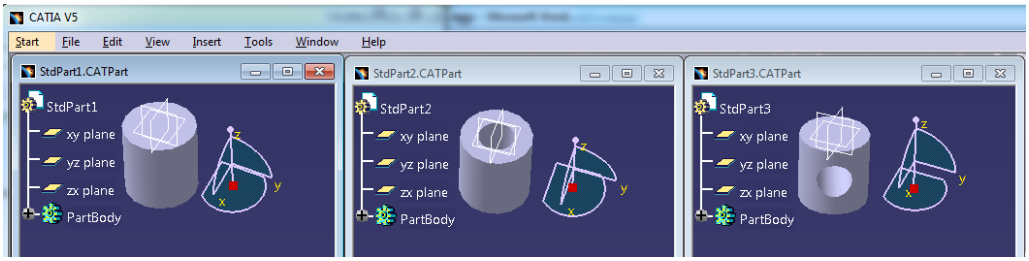
As in the part structure mode standard part items are treated differently than regular CAD document items. Standard part CATParts (CATProducts are not allowed) are defined by two things:

1. The PDM attribute ‘is_standard’ (“Standard Part”) is set to ‘1’.
2. In the CATPart there is a user-defined CATIA property with the name ‘IsStandardPart’ and the value of ‘true’.



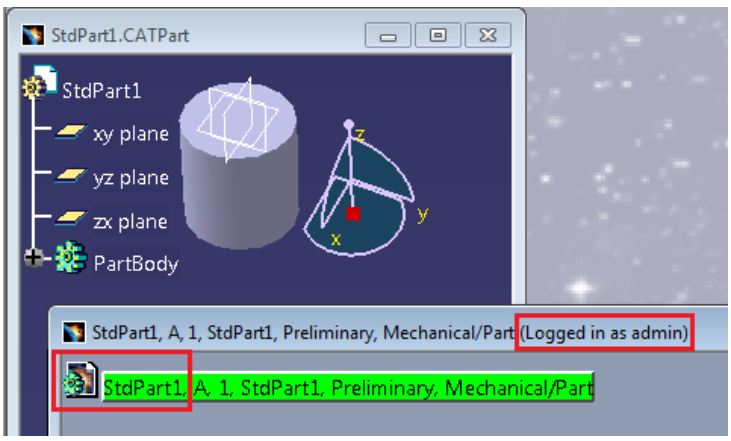
Picture 47: “IsStandardPart” user-defined property

In this example there are three CATParts which are defined as standard parts: “StdPart1.CATPart”, “StdPart2.CATPart”, and “StdPart3.CATPart”:



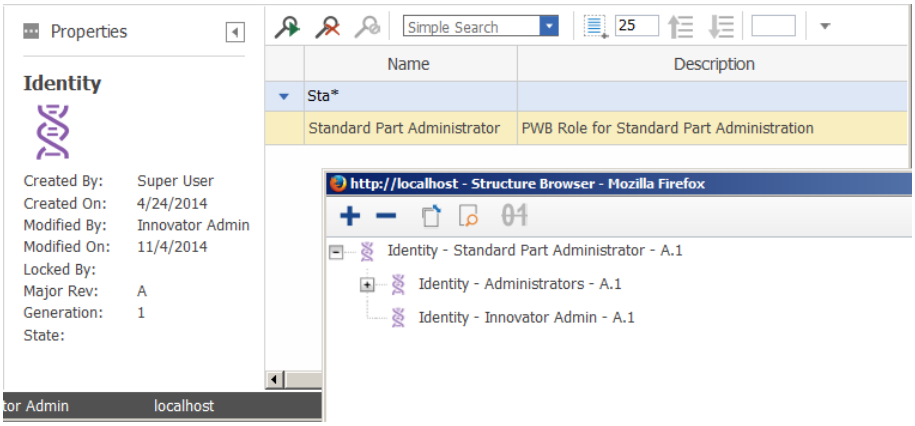
Picture 48: Three example standard part CATParts

Standard part items can only be created by a user which is defined as standard part administrator.



Picture 49: User is logged in as administrator

By default “Innovator Admin” is a standard part administrator, that is, a member of the identity “Standard Part Administrator”:



Picture 50: The Standard Part Administrator identity

Management of CATIA templates

TemplateFileAdmin

The PWB Configuration item setting “TemplateFileAdmin” defines the template file administrator identity.

Mandatory. Can be removed if the following server events are removed:

PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files

Server configuration

CAD documents with the attribute ‘is_template’ having the value ‘1’ and their related file items have to be imported to PDM. This can be done with a batch process, or with the PDM Workbench integration itself.

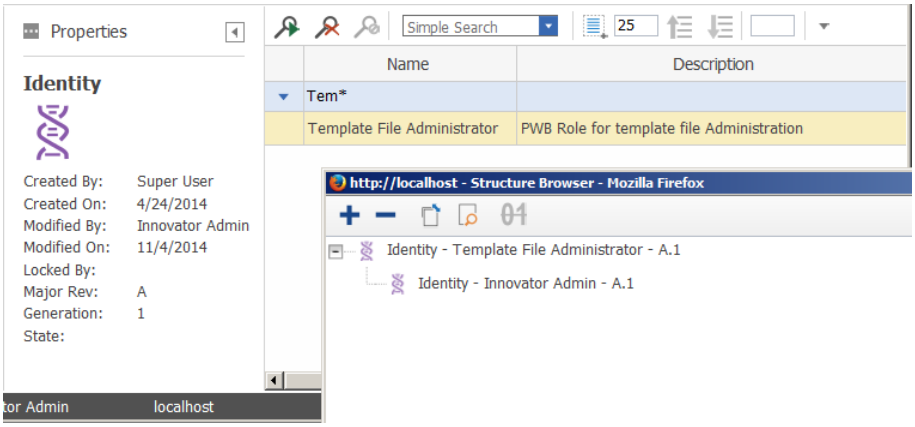
For this, the user has to be in a special template file administrator group.

This group can be configured as follows:

The configuration variable “TemplateFileAdmin” in the “Settings” tab of the active PWB configuration item has to contain the name of an identity as its value, for instance “Template File Administrator”:

TemplateFileAdmin	Template File Administrator

Picture 51: Template file administrator configuration variable



Picture 52: The Template File Administrator identity

If this identity contains the user which is currently logged in then the user is a template file administrator and has the rights to create and modify template file documents.

For regular users existing template file documents are read-only (black text background).

One way of creating template file documents in Innovator with PDM Workbench is this:

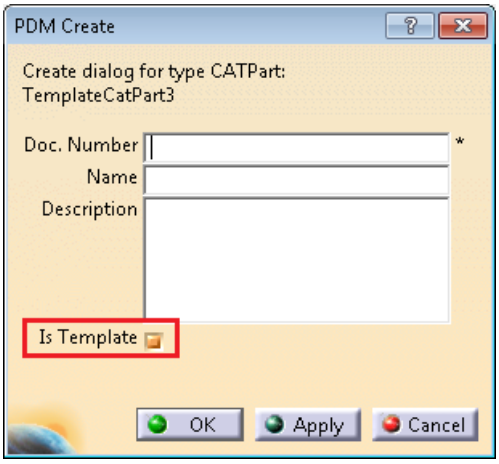
1. Make sure the following settings are set in the active PWB configuration item, at least temporarily:

UseBomPartStructure	false
ShowCreateDialogsDuringUpdate	true

Picture 53: PWB Configuration variables for template creation

2. Log in to Innovator with PDM Workbench as a template file administrator.
3. Open a CATIA file which is supposed to be used as a PDM template document in CATIA V5.
4. Click on the Update icon in the PDM Workbench toolbar to create the CAD document and to upload the file.

Since the two settings are set to the values described in step 1 only a CAD document will be created, and a create dialog is shown to the user:

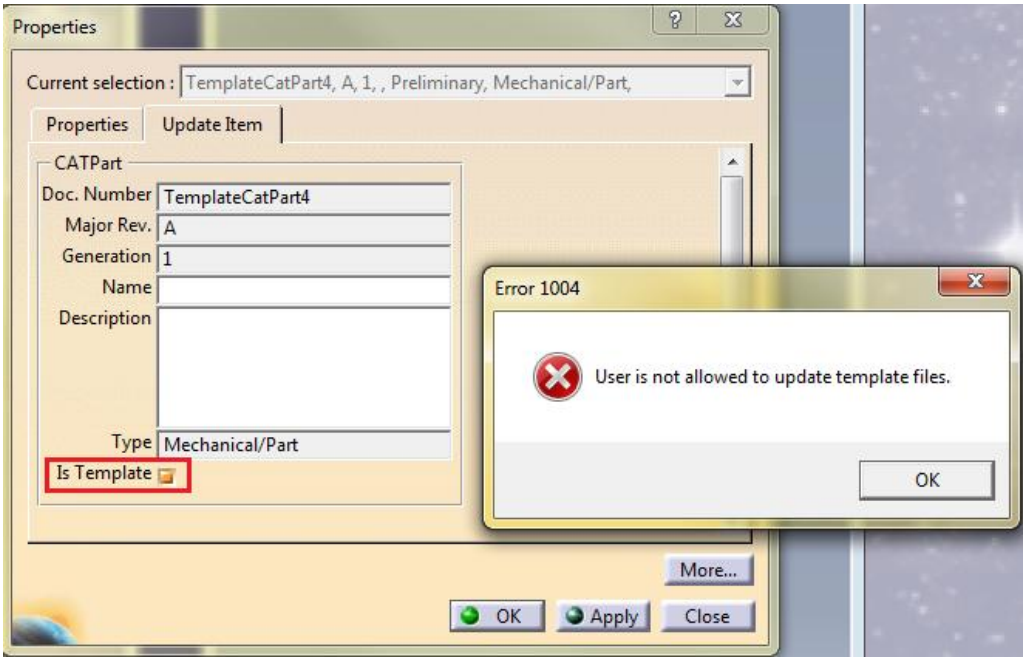


Picture 54: Create dialog containing "Is Template" checkbox

The document number has to be entered, and it is important that the "Is Template" attribute is checked.

Another possibility would be to first create the template file as a regular file, and to change the “is_template” attribute afterwards. In this case the “ShowCreateDialogsDuringUpdate” setting does not have to be set to “true”. Even the “UseBomPartStructure” setting can be set to “true”, but in this case a Part item is created, too. This Part item may have to be deleted later, because it should probably not be in the bill of materials.

Only a template file administrator can change the “is_template” attribute value of a CAD document item. A regular user will get this error message:



Picture 55: Template file creation error message

For a regular user template file CAD document items are read-only and they also can not be locked by a regular user.

Client configuration

In the schema file the configuration ‘templateFiles’ has to exist and it has to contain the attribute ‘ loadFrom="PDM" ’:

```
<templateFiles loadFrom="PDM"
               catPrt="PwbCatPrtTmplFileNames"
               catPrd="PwbCatPrdTmplFileNames"
               catDrw="PwbCatDrwTmplFileNames" />
```

The values of the XML attibutes ‘catPrt’, ‘catPrd’, and ‘catDrw’ have to contain the names of data source definitions of the type ‘ValueList’ which contain the file names of the CATPart, CATProduct, and CATDrawing template files:

```
<dataSource name="PwbCatPrtTmplFileNames" type="ValueList">
      <value name="TemplateCatPart1.CATPart" />
      <value name="TemplateCatPart2.CATPart" />
</dataSource>

<dataSource name="PwbCatPrdTmplFileNames" type="ValueList">
      <value name="TemplateCatProduct1.CATProduct" />
      <value name="TemplateCatProduct2.CATProduct" />
</dataSource>
```

```
<dataSource name="PwbCatDrwTmplFileNames" type="ValueList">
      <value name="TemplateCatDrawing1.CATDrawing" />
      <value name="TemplateCatDrawing2.CATDrawing" />
</dataSource>
```

The first file in each of these lists is the default template, and that template file is used when the template file functionality is switched on, and if a part structure with no related CAD documents is loaded to CATIA.

The old template file functionality, where locally accessible files are used, is turned on by a definition like this in the schema file:

```
<templateFiles loadFrom="C:\Users\Public\PDM-Workbench\Templates"
               catPrt="PwbCatPrtTmplFileNames"
               catPrd="PwbCatPrdTmplFileNames"
               catDrw="PwbCatDrwTmplFileNames" />
```

In this definition the absolute file path which is defined as the value of the “loadFrom” XML attribute has to contain the CATIA files which are defined in the data sources which are referenced in the XML attributes “catPrt”, “catPrd”, and “catDrw”.

If ‘loadFrom=”PDM”‘ is set, then CAD documents with the same document number (without the file extension) and the PDM attribute “is_template” having the value “1” have to be defined in Innovator, and they have to be related to File items with the same file name (with the file extension).

Additionally the “Create” dialog definitions for the PDM types for CATPart, CATProduct, and CATDrawing files need to contain the form attribute for the template.

Here an example is given for the CATPart type:

```
<form name="Create">
      <formAttribute name="CatPrtTemplate" widgetType="ComboBox"
                     mode="update" visibleLength="30" required="false" />
</form>
```

The referenced attribute with the name “CatPrtTemplate” needs to contain two specific definitions: A data source which contains the names of the template files (as described above), and a specific information that this attribute contains information about CATPart template files (shown in red below).

```
<attribute name="CatPrtTemplate" displayName="NLS_CatPrtTemplate"
           dataSource="PwbCatPrtTmplFileNames"
           pwbAttrInfo="CatPrtTemplateListAttr"/>
```

When the template files for one of the three types should not be used, e.g. no template support for CATDrawings, then the value “catDrw” has to be removed from the “templateFiles” tag and the form attribute for the template has to be removed from the create dialog.

Thumbnails

The PWB Configuration item setting “CreateThumbnailsFromTypes” is a list of file extensions for which the thumbnails should be created by update, separated by "|".

CreateThumbnailsFromTypes	.CATPart|.CATDrawing

Picture 56: Sample CreateThumbnailsFromTypes configuration

Default value: .CATPart|.CATDrawing

Optional.

Derived viewable files

PDF files derived from CATDrawings, and 3DXML files derived from CATParts can optionally be generated and uploaded at every Update.

In the active PWB Configuration item the settings “UpdateDerivedPdfFile” and “UpdateDerived3DxmlFile” have to be set to “true”:

UpdateDerived3DxmlFile	true
UpdateDerivedPdfFile	true

Picture 57: Sample Derived Viewable Files configuration

Archives

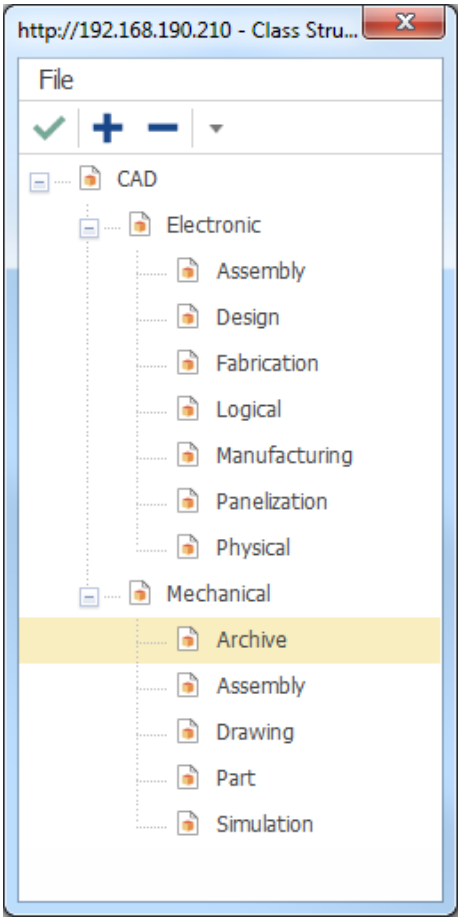
It is possible to compress a complete CATProduct sub-structure into one Zip file and to manage this compressed file in PDM. This makes it possible to hide a complicated CATProduct structure in one CAD document if it is not necessary to manage the structure information in PDM.

The server-side configuration setting “UseArchives” has to be set to “true”:

UseArchives	true

Picture 58: Sample UseArchives configuration

If archives are used a new classification for the item type CAD needs to be defined, for instance “Mechanical/Archive”.



Picture 59: Class structure for type 'CAD' – Archive added

This classification has to be the same as the one defined in the PWBSchema.xml client configuration file, e.g.:

```
<object name="/CAD/Mechanical/Archive"
        displayName="NLS_Archive"
        icon="Aras_Archive">
```

Design tables

The PWB Configuration item setting "UseDesignTables" configures the use of Design Tables with the PDM Workbench. The use of Design Tables in Aras Innovator can be enabled or disabled in PDM Workbench.

Please note that when Design Tables are disabled from PDM Workbench, they can still be used in CATIA, but they are not synchronized and uploaded to Aras Innovator.

UseDesignTables	true

Picture 60: Sample UseDesignTables configuration

Default value: false

Optional. Possible values: "true", or "false".

Add Temp configuration

In the Schema file the prefix for the rename of the Part Numbers and File Names for the "Add Temp" and "Open File Temporary" command can be defined.

Example:

```
<addTempPrefix value="TMP" />
```

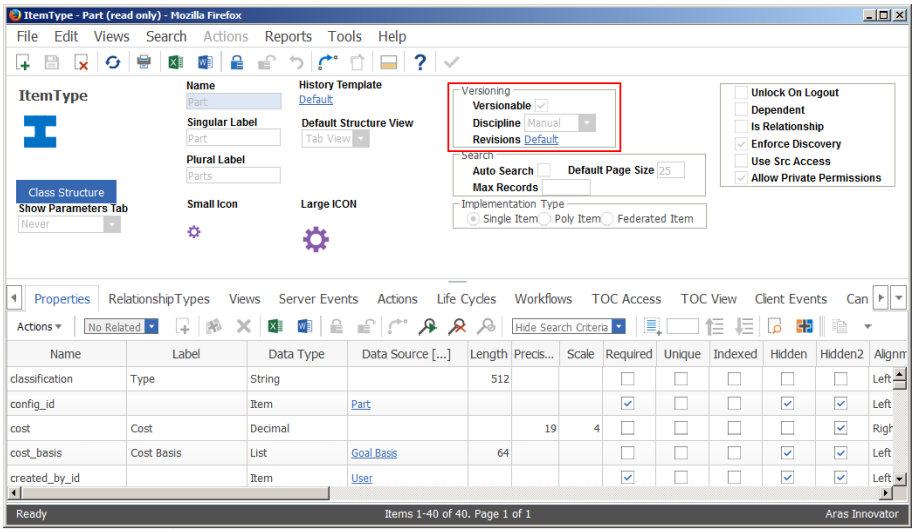
If the add temp prefix value is defined by the environment variable "PWB_ADDTEMP_PREFIX", then this one takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

Default value: TMP

Optional.

Versioning

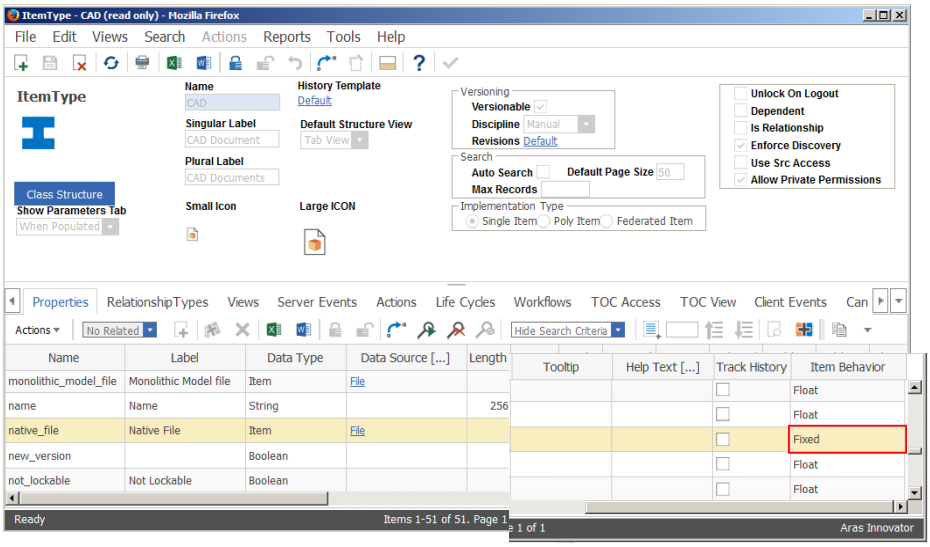
Please set the versioning discipline for the item types "Part" and "CAD" to "Manual". Now the new generation of a part and CAD document will not be created automatically in case of an update (see *Picture 61: Item Type "Part"*).



Picture 61: Item Type "Part"

Accessibility of old file versions

Please set the "Item Behavior" of the property "native_file" for the item type "CAD" to "Fixed". Now the old file versions can be accessed (see *Picture 62: Item Type "CAD"*).



Picture 62: Item Type "CAD"

Non-BOM CATParts and CATProducts

In the part structure mode it is now possible to define CATParts and CATProducts in the CATIA structure to be defined as not BOM-relevant. In this case no corresponding part items will be created in PDM.

The server-side configuration setting “UseNonBom3DCadDocs” has to be set to ‘true’:

UseNonBom3DCadDocs	true

Picture 63: Sample UseNonBom3DCadDocs configuration

Check CAD Links

When CATIA documents with 3D links need to be imported this functionality helps the user to determine which documents have to be imported in which order, and which documents have to be in the CATIA session so the links are created correctly.

To retrieve the CAD documents which already exist in PDM by default the CAD document number is compared to the CATIA file name. If the CAD documents in PDM have been assigned a PDM-generated number then the original file name has been overwritten. If the original file name has been saved in another PDM attribute this other attribute can be queried instead.

If, for instance, the original file name has been saved in the ‘name’ attribute of the CAD document, then this functionality can be configured to query for the ‘name’ attribute when comparing the value with the CATIA file name.

For this the PWB Configuration item setting "CadLinkCheckQueryAttribute" can be set to a different PDM attribute, e.g. “name”.

CadLinkCheckQueryAttribute	name

Picture 64: Sample CadLinkCheckQueryAttribute configuration

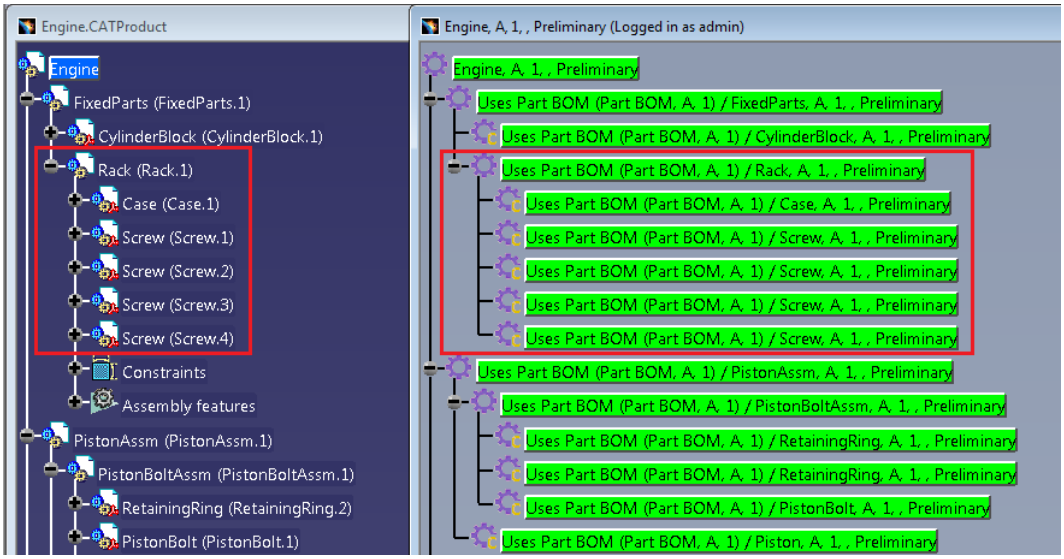
Displaying part structure instances as separate nodes

The display of the part structure in the PDM structure window can be changed such that every part instance is shown as a separate node.

The PWB Configuration item setting “UseSeparateRelationsForInstances” has to be set to “true” to switch on this functionality.

UseSeparateRelationsForInstances	true

Picture 65: Sample UseSeparateRelationsForInstances configuration


Engine.CATProduct
Engine
FixedParts (FixedParts.1)
CylinderBlock (CylinderBlock.1)
Rack (Rack.1)
Case (Case.1)
Screw (Screw.1)
Screw (Screw.2)
Screw (Screw.3)
Screw (Screw.4)
Constraints
Assembly features
PistonAssm (PistonAssm.1)
PistonBoltAssm (PistonBoltAssm.1)
RetainingRing (RetainingRing.2)
PistonBolt (PistonBolt.1)
Engine, A, 1, , Preliminary (Logged in as admin)
Engine, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / FixedParts, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / CylinderBlock, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / Rack, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / Case, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / Screw, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / Screw, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / Screw, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / Screw, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / PistonAssm, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / PistonBoltAssm, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / RetainingRing, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / RetainingRing, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / PistonBolt, A, 1, , Preliminary
Uses Part BOM (Part BOM, A, 1) / Piston, A, 1, , Preliminary

Picture 66: Every part instance is shown as separate node

Support for the new CAD structure instance handling introduced in Innovator 9.4 and 10.0

A new relation with the name “CAD Instance” has been introduced, which contains instance information for “CAD Structure” relations.

By default the new CAD Instance relation is used to store CATIA instance information.

If the setting “UseCadInstance” in the used PWB Configuration item is set to “false” then the old instance handling (using three attributes on “CAD Structure” to store the instance name, the instance description, and the transformation matrix) is used.

UseCadInstance	true

Picture 67: Sample UseCadInstance configuration

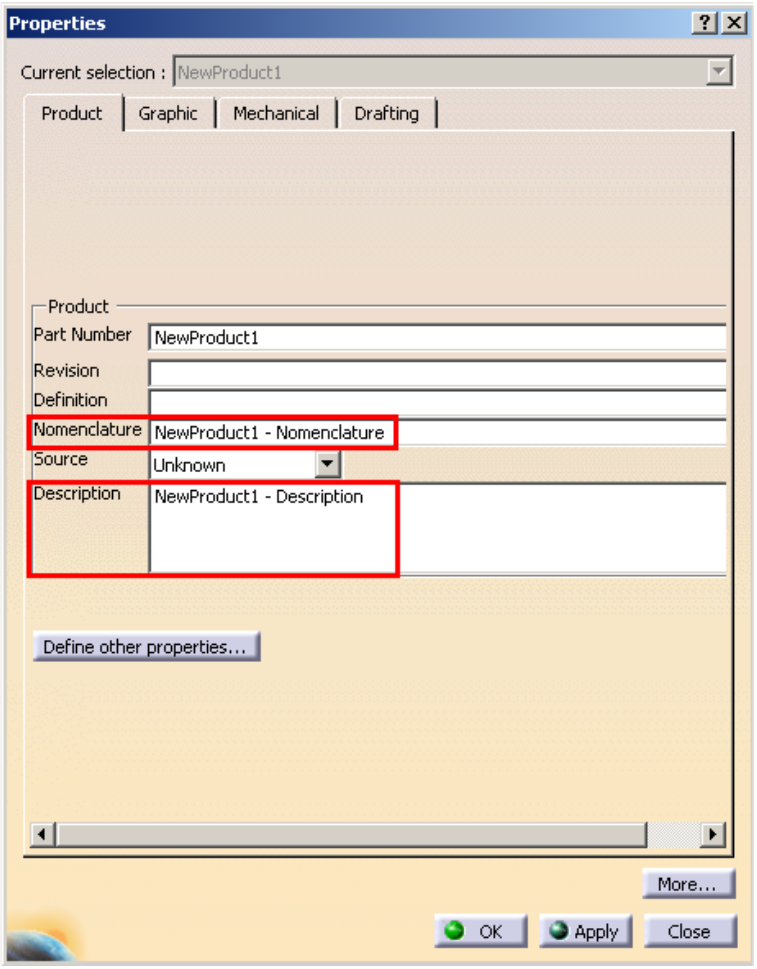
Attribute Mapping Configuration

Standard attribute mapping

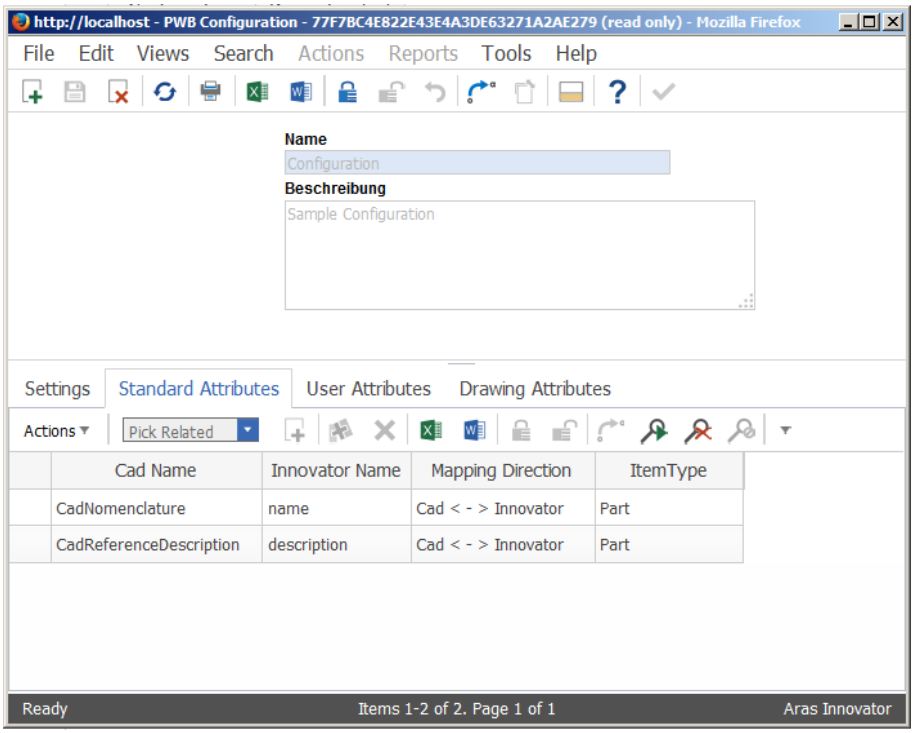
Here is a description of how to configure the attribute mapping:

CATIA standard and user-defined properties can be mapped to PDM attributes.

In the following example the standard CATIA attributes “Nomenclature” and “Description” are mapped to the attributes “name” and “description” of the Aras Innovator part object (see *Picture 68: Standard attributes in the “Properties” dialog* and *Picture 69: Configuration of standard attributes in Aras Innovator*).

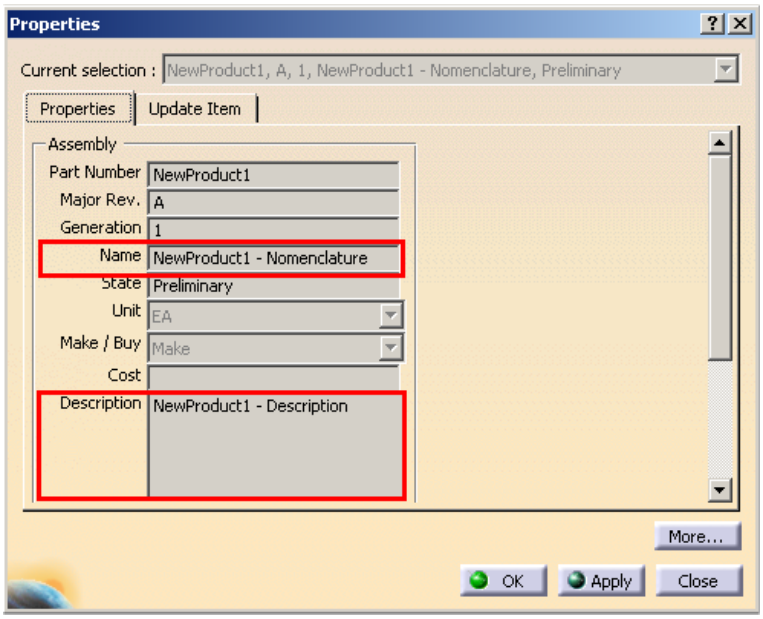


Picture 68: Standard attributes in the “Properties” dialog

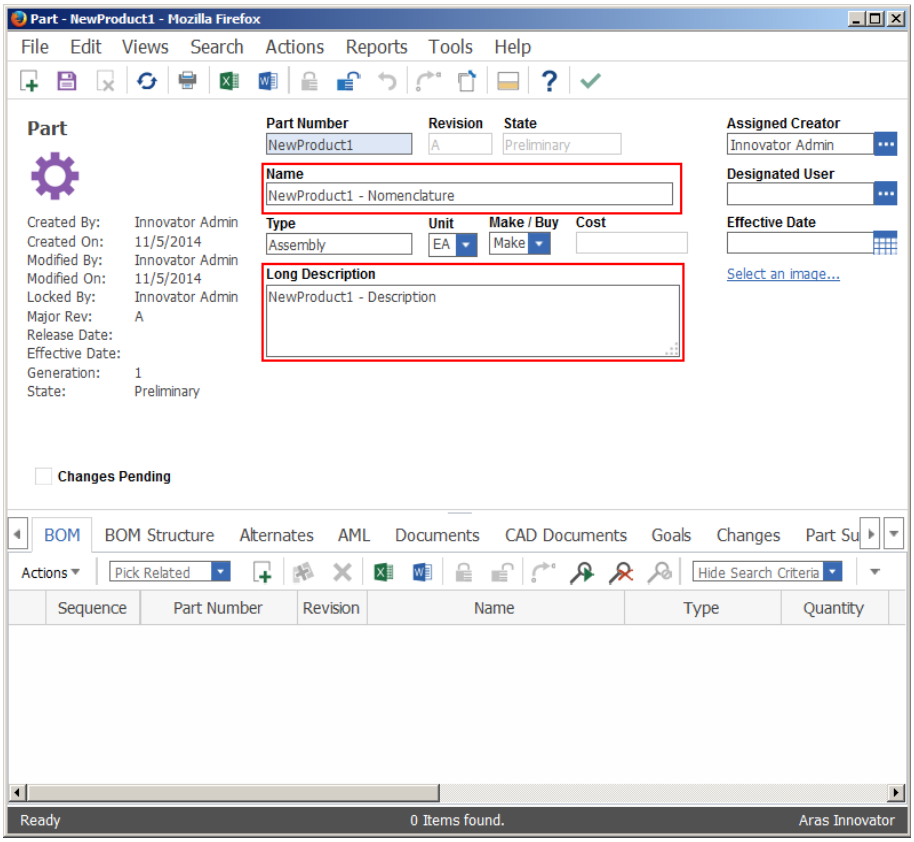

http://localhost - PWB Configuration - 77F7BC4E822E43E4A3DE63271A2AE279 (read only) - Mozilla Firefox
File Edit Views Search Actions Reports Tools Help
Name
Configuration
Beschreibung
Sample Configuration
Settings Standard Attributes User Attributes Drawing Attributes
Actions Pick Related
Cad Name Innovator Name Mapping Direction ItemType
CadNomenclature name Cad < - > Innovator Part
CadReferenceDescription description Cad < - > Innovator Part
Ready Items 1-2 of 2. Page 1 of 1 Aras Innovator

Picture 69: Configuration of standard attributes in Aras Innovator

After creating the part with Update the defined CATIA attribute values have been written to the PDM part object (see *Picture 70: Standard attributes in the "Properties" dialog of the PDM node* and *Picture 71: Standard attributes in Aras Innovator window*).



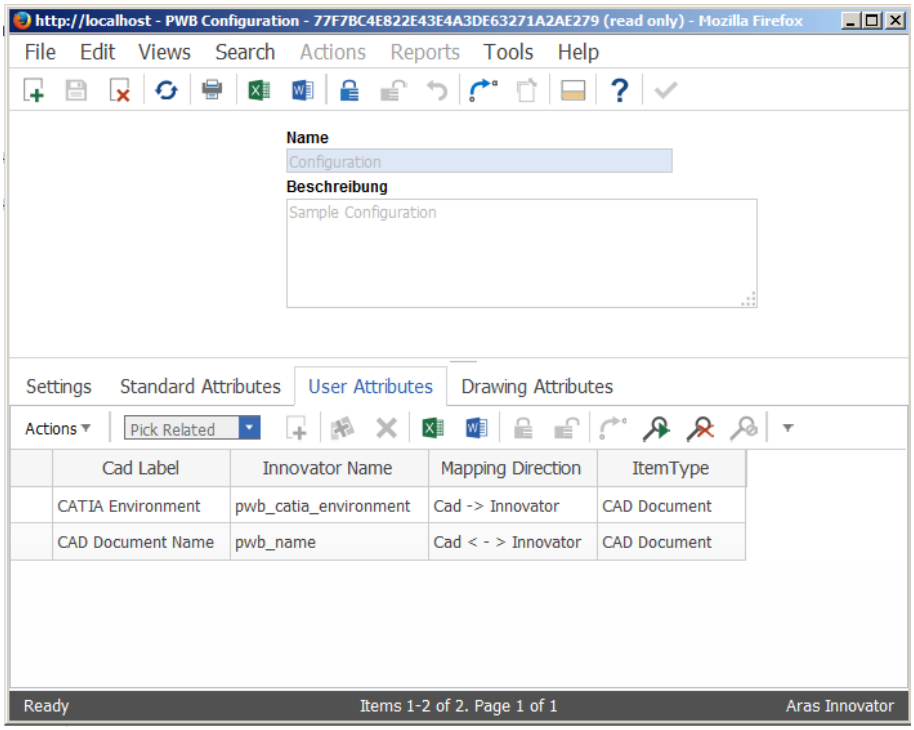
Picture 70: Standard attributes in the "Properties" dialog of the PDM node



Picture 71: Standard attributes in Aras Innovator window

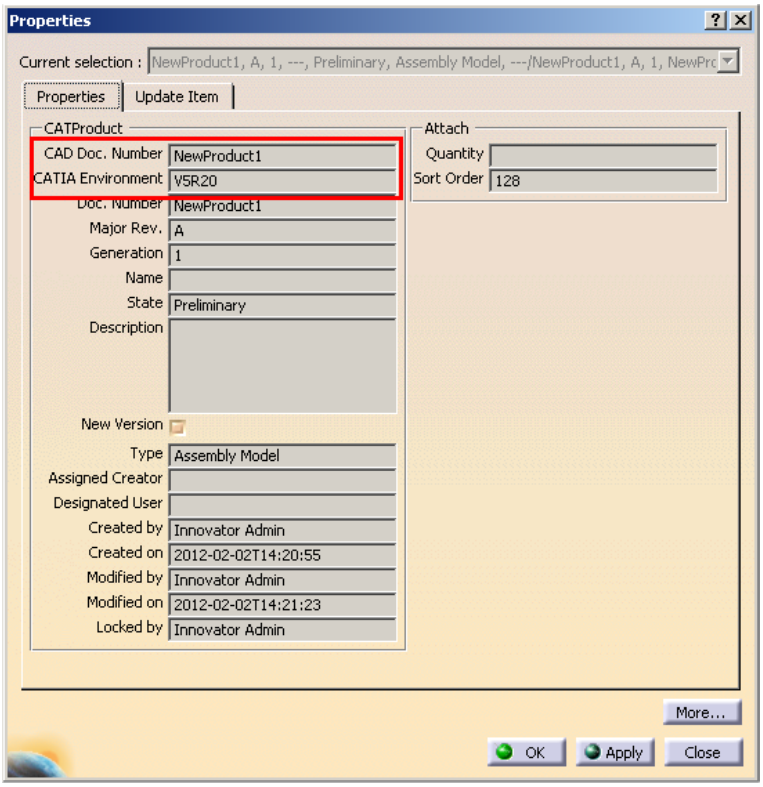
User attribute mapping

User-defined CATIA properties can also be mapped (see *Picture 72: Configuration of user-defined attributes in Aras Innovator*).

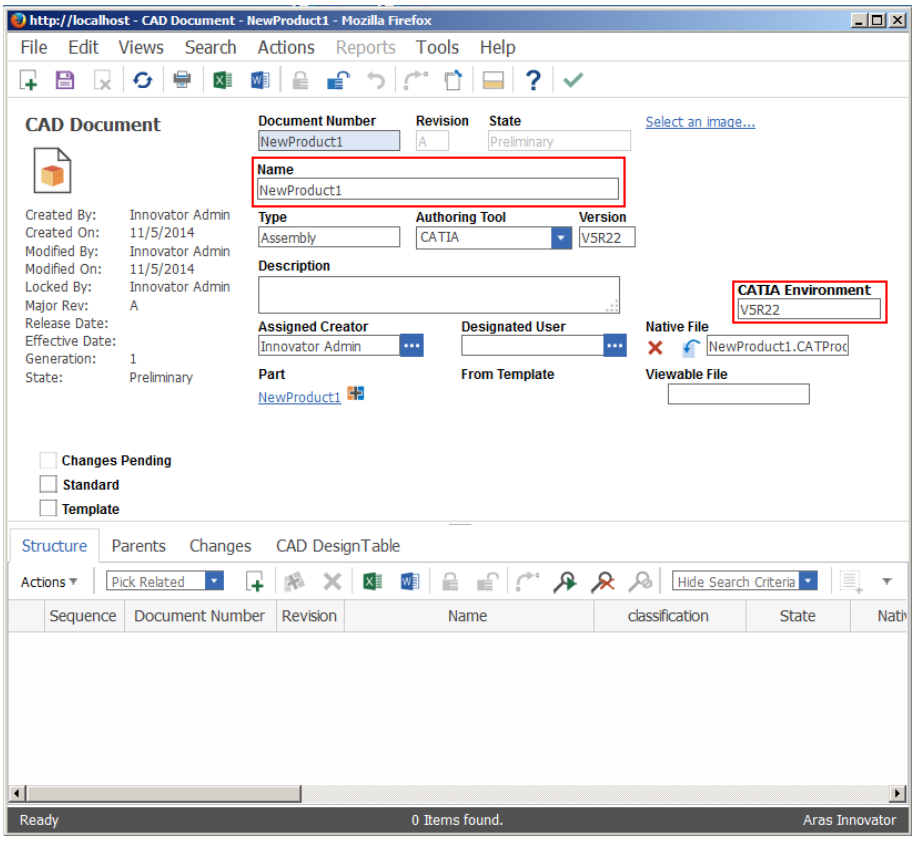


Picture 72: Configuration of user-defined attributes in Aras Innovator

While the structure is imported the values are written to the defined attributes of the Aras Innovator CAD document object (see and *Picture 73: User-defined attributes in the "Properties" dialog of the PDM node* and *Picture 74: User-defined attributes in Aras Innovator window*).

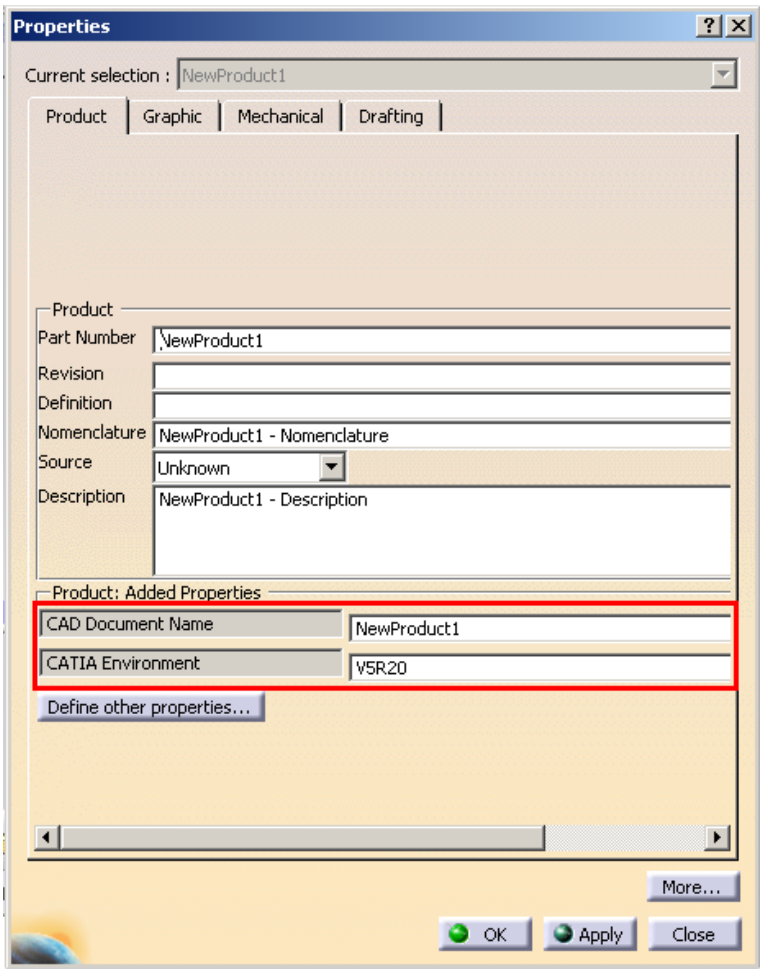


Picture 73: User-defined attributes in the "Properties" dialog of the PDM node



Picture 74: User-defined attributes in Aras Innovator window

After the import or after loading the structure it can be shown that the values are written from the PDM attributes into the CATIA files (see *Picture 75: User-defined attributes in the "Properties" dialog*).



Picture 75: User-defined attributes in the "Properties" dialog

CATIA user defined attributes

Several user defined properties can be filled by CATIA. These properties have to be defined in the Schema file. During the update process in CATIA the property values are filled.

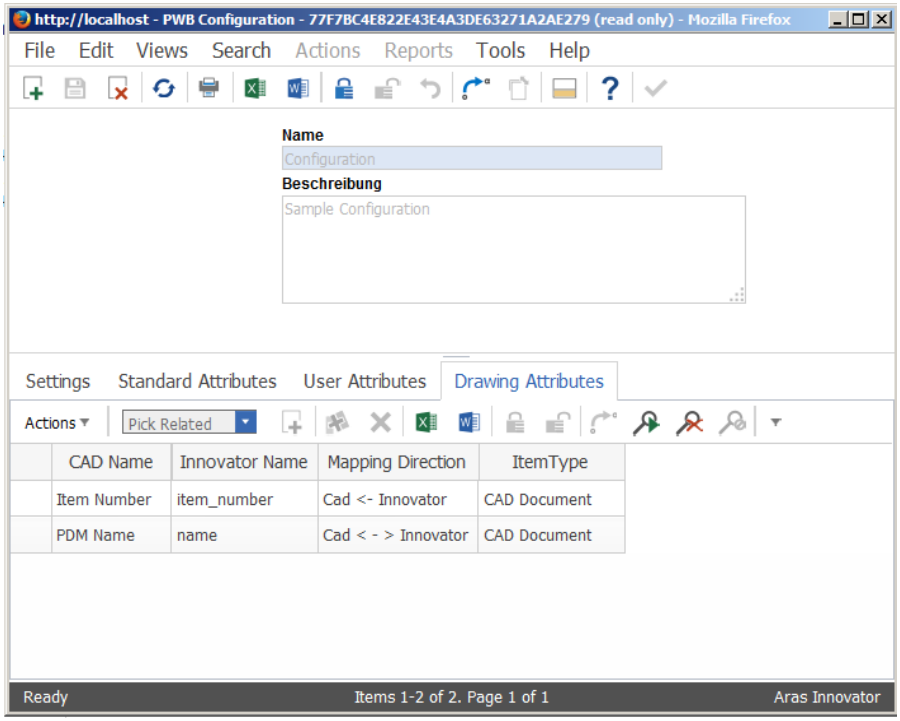
Example:

```
<updateCatiaUserDefinedProperties>

   <userDefinedProperty catiaInfo="CatiaLevel"
                        propertyDisplayName="CATIA Environment"/>

   <userDefinedProperty catiaInfo="Mass"
                        propertyDisplayName="The Mass"/>

   <userDefinedProperty catiaInfo="Volume"
                        propertyDisplayName="The Volume"/>

   <userDefinedProperty catiaInfo="Density"
                        propertyDisplayName="The Density"/>

   <userDefinedProperty catiaInfo="Area"
                        propertyDisplayName="The Area"/>

</updateCatiaUserDefinedProperties>
```

When the values should to be sent to Aras they have to be mapped before. For details see the section above.

Drawing attribute mapping

The configuration of the drawing attribute mapping has changed. The drawing attributes are defined like the 3D geometry attributes now.

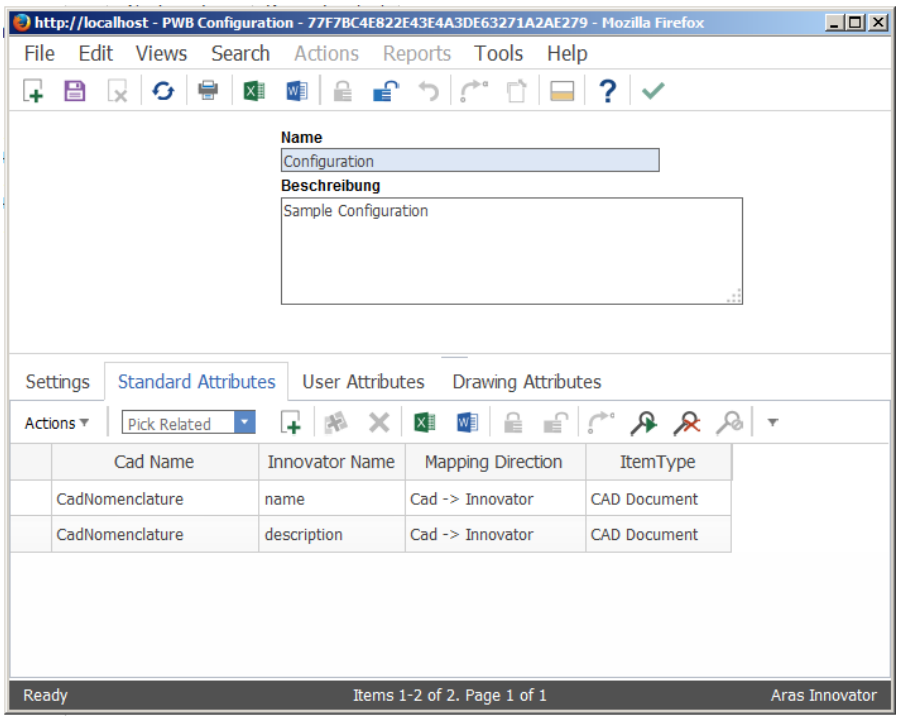

http://localhost - PWB Configuration - 77F7BC4E822E43E4A3DE63271A2AE279 (read only) - Mozilla Firefox
File Edit Views Search Actions Reports Tools Help
Name
Configuration
Beschreibung
Sample Configuration
Settings Standard Attributes User Attributes Drawing Attributes
Actions
Pick Related
CAD Name Innovator Name Mapping Direction ItemType
Item Number item_number Cad <- Innovator CAD Document
PDM Name name Cad < - > Innovator CAD Document
Ready
Items 1-2 of 2. Page 1 of 1
Aras Innovator

Picture 76: Configuration of drawing attributes in Aras Innovator

Extended attribute mapping functionality

It is possible to define multiple assignments for the same CATIA property in the CAD to PDM direction.

The attribute mapping in the PWB Configuration item can contain multiple PDM attributes for the same CATIA attribute.


http://localhost - PWB Configuration - 77F7BC4E822E43E4A3DE63271A2AE279 - Mozilla Firefox
File Edit Views Search Actions Reports Tools Help
Name
Configuration
Beschreibung
Sample Configuration
Settings Standard Attributes User Attributes Drawing Attributes
Actions
Pick Related
Cad Name Innovator Name Mapping Direction ItemType
CadNomenclature name Cad -> Innovator CAD Document
CadNomenclature description Cad -> Innovator CAD Document
Ready
Items 1-2 of 2. Page 1 of 1
Aras Innovator

Picture 77: Example PWB Configuration attribute mapping

Basic Drawing Link Support

The drawing link functionality has to be switched on by defining the schema file setting

```
<updateCatiaLinksInPdm value="true" />
```

In addition to that, an additional form attribute has to be defined in the “Register” form of the CATDrawing object definition. This attribute has to have the attribute displayOnly=”true” to indicate that it is not an actual PDM attribute, only a display attribute in the dialog:

```
<formAttribute name="LinkedMainGeometryFile" widgetType="ComboBox"
               mode="update" visibleLength="15" required="false"
               listViewRelevant="false" displayOnly="true" />
```

The corresponding attribute definition has to contain the XML attribute pwbAttrInfo="MainDrwGeometryFile", and it has to refer to a data source which contains some special definitions:

```
<attribute name="LinkedMainGeometryFile"
           displayName="NLS_LinkedMainGeometryFile"
           dataSource="LinkedGeometryFiles"
           pwbAttrInfo="MainDrwGeometryFile" />
```

The data source definition which is referred has to contain the definition additionalValues="GeometryFilesLinkedInSession":

```
<dataSource name="LinkedGeometryFiles" type="ValueList"
            additionalValues="GeometryFilesLinkedInSession" />
```

If the configuration is set up correctly then the user will see an additional list widget in the register dialog for CATDrawings which enables him to select one of the linked 3D geometry files (see *PDM Workbench User Manual*).

Download related 3D File

In the Schema file it can be configured if with the loading of the drawing file the reference 3D file should also be downloaded and opened in CATIA.

Example:

```
<downloadDrawingRelated3DFile value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Duplicate related drawings

When a CATPart or a CATProduct is duplicated with the PDM Workbench, the related CATDrawing will also be duplicated if the part number of the CATPart/CATProduct is the prefix of the CATDrawing, e.g. “Part1Drw” is related to “Part1” and has the correct prefix.

In the Schema file the length of the prefix to be compared can be defined.

Example:

```
<duplicateRelatedDrawings charsToCompare="25" />
```

Default value: 0

Optional.

Basic Multi-Model Link Support

In the Schema file the multi-model link functionality can be switched on, similar to the drawing link functionality.

Example:

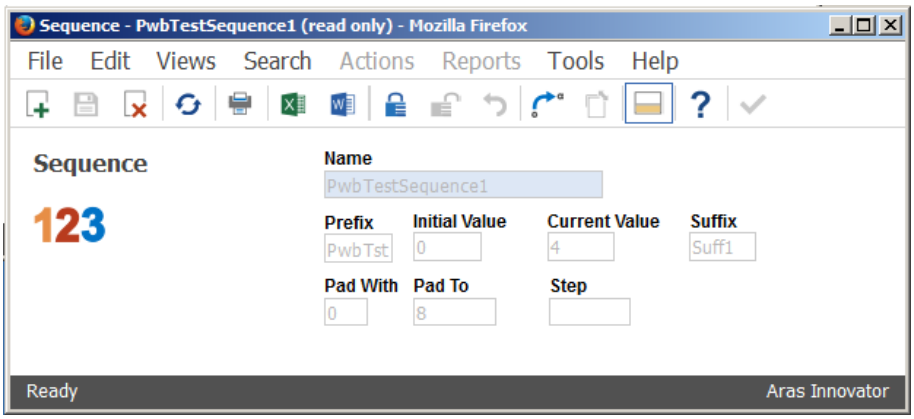
```
<updateCatiaLinksInPdm value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

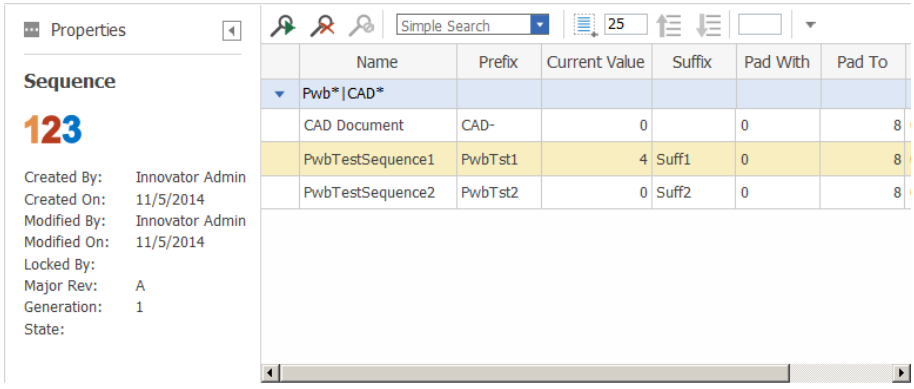
Autoname Support using Innovator Sequence Items

First sequence Innovator items which should be used for the autoname functionality need to be created:



Picture 78: Sample Sequence item

The following sequence items will be used in the configuration example:



Picture 79: Sequence items used in example

Then the sequence items to be used need to be configured in the schema file:

First an attribute with a data source which contains the names of the sequence items needs to be defined:

```
<attribute name="pwbAutonameRule" displayName="NLS_AutonameRule"
           dataSource="AutonameRules" />
```

```
<dataSource name="AutonameRules" type="ValueList">
      <value name="PwbTestSequence1" displayName="" />
      <value name="PwbTestSequence2" displayName="" />
      <value name="CAD Document" displayName="" />
</dataSource>
```

Then a corresponding form attribute has to be included in the login dialog …

```
<form name="Login" info="ShowOnlyLoginData" >
      <frame displayName="NLS_UserData">
            ...
            <formAttribute name="pwbAutonameRule"
                           widgetType="ComboBox"
                           mode="update" visibleLength="15"
                           required="false" entryAllowed="false" />
      </frame>
</form>
```

… and in the “Set PDM Configuration” dialog.

```
<form name="PdmSessionConfig">
      <formAttribute name="pwbAutonameRule" widgetType="ComboBox"
                     mode="update" visibleLength="15" required="false"
                     listViewRelevant="true" />
</form>
```

This will enable the user to select a sequence item name as an autoname rule either at login or later while working in the PDM Workbench session.

In order for the “Set PDM Configuration” dialog to appear the setting "SetSessionConfig" has to be removed from the “removeToolbarIcons” definition:

```
<removeToolbarIcons>
      <!-- "Register", "Update", "Synchronize", "Refresh",
           "SetSessionConfig", "NewPwbWindow", "DynamicActions", "About"
        -->
      <icon name="Register" />
      <icon name="Synchronize" />
      <icon name="NewPwbWindow" />
      <!-- <icon name="SetSessionConfig" /> -->
      <icon name="DynamicActions" />
</removeToolbarIcons>
```

Autoname functionality can use a server method

The autoname functionality can use a server method instead of using a sequence item directly for obtaining a PDM-generated part or document number value.

The autoname functionality has to be configured in the PWBSchema.xml file.

The setting “UseServerMethodsForAutoname” has to be set to “true” in the active PWB Configuration item.

Additionally server methods whose names correspond to the names configured in the PWBSchema.xml file has to be defined on the Innovator server.

The server method can use information from standard CATIA attributes of the CATIA files to be imported to PDM, or values from PDM Workbench dialogs.

This is an example of such a server method:

```
ArasUtil ArasUtilObj = new ArasUtil();
Innovator InnovatorObj = this.getInnovator();

// Preparing the input information
string Autoname = this.getProperty("Autoname");
string PdmType = this.getProperty("Type");
string PdmClassification = this.getProperty("Classification");

Item CatiaStdPropsItem = this.getPropertyItem("CatiaStdProps");
IDictionary<string, string> CatiaStdPropsDict = null;
if (CatiaStdPropsItem != null)
{
    CatiaStdPropsDict =
       ArasUtilObj.DialogAttrsItemToDictionary(CatiaStdPropsItem);
}

Item CadDocInputDialogItem = this.getPropertyItem("CadDocDialogAttrs");
IDictionary<string, string> CadDocInputDialogDict = null;
if (CadDocInputDialogItem != null)
{
    CadDocInputDialogDict =
       ArasUtilObj.DialogAttrsItemToDictionary(CadDocInputDialogItem);
}

Item PartInputDialogItem = this.getPropertyItem("PartDialogAttrs");
IDictionary<string, string> PartInputDialogDict = null;
if (PartInputDialogItem != null)
{
    PartInputDialogDict =
       ArasUtilObj.DialogAttrsItemToDictionary(PartInputDialogItem);
}

// Place the custom algorithm here
string OutputLogInfo = "";

OutputLogInfo += "Autoname:'" + Autoname + "'" + "|";
OutputLogInfo += "PdmType:'" + PdmType + "'" + "|";
OutputLogInfo += "PdmClassification:'" + PdmClassification + "'" + "|";

if (CatiaStdPropsDict != null)
{
    OutputLogInfo += "CatiaStdProps:" + "|";
    var Enumerator = CatiaStdPropsDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "'" + CurrentVal.Key + "'->'" +
               CurrentVal.Value + "'" + "|";
    }
}

if (CadDocInputDialogDict != null)
{
       OutputLogInfo += "CadDocInputDialog:" + "|";
    var Enumerator = CadDocInputDialogDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "'" + CurrentVal.Key + "'->'" +
               CurrentVal.Value + "'" + "|";
    }
}

if (PartInputDialogDict != null)
{
       OutputLogInfo += "PartInputDialog:" + "|";
    var Enumerator = PartInputDialogDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "'" + CurrentVal.Key + "'->'" +
```

```
                CurrentVal.Value + "'" + "|";
        }
}

// Getting the actual autoname value
string AutonameValue =
        ArasUtilObj.GetNextAutonameSequence(InnovatorObj, Autoname);

OutputLogInfo += "done";

// Preparing the output
IDictionary<string, string> OutputInfoDict = new Dictionary<string, string>();

OutputInfoDict.Add("AutonameValue", AutonameValue);

OutputInfoDict.Add("LogLines", OutputLogInfo);

Item OutputInfoItem = ArasUtilObj.DialogAttrsDictionaryToItem(InnovatorObj,
OutputInfoDict);

return OutputInfoItem;
```

Configurable CATIA components support

It is possible to load and update CATProduct structures which contain embedded CATIA components. Depending on the part number prefix the embedded component nodes can either be “skipped”, that is, the node is treated as if it does not exist, but its child nodes are processed, or they can be “ignored”, that is, the node and all its child nodes are treated as if they do not exist.

The configurable node behavior functionality has to be switched on by defining the schema file setting “catiaNodeBehaviorDefinitions”:

```
<catiaNodeBehaviorDefinitions>

        <catiaNodeBehavior catiaNodeType="EmbeddedComponent"
                           partNumberPrefix="SKIP_" behavior="SkipNode"/>

        <catiaNodeBehavior catiaNodeType="EmbeddedComponent"
                           partNumberPrefix="IGN_" behavior="IgnoreNode"/>

</catiaNodeBehaviorDefinitions>
```

The behavior settings “SkipNode” and “IgnoreNode” are allowed.

Support for relating a new CATIA file to an existing Part

The currently active CATIA document (only CATParts or CATDrawings) can be related to an existing BOM part item. If there is already a corresponding CAD document related to the part the document’s file can be overwritten.

For part items the context action “RelateToPart” can be added in the Schema file:

```
<contextAction name="RelateToPart" usedIn="QueryDialog" />
```

“Delete relation” context menu action in the PDM structure window

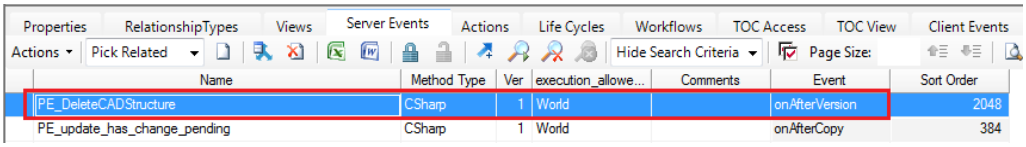
PDM relations can be deleted in the PDM structure window with a single context menu action now, even if the PDM relations are not displayed in the structure.

For any item the context action “DeleteRelation” can be added in the Schema file:

```
<contextAction name="DeleteRelation" usedIn="PdmWindow" />
```

The PE_DeleteCADStructure server event method can be removed

The PE_DeleteCADStructure server event method, which is used for deleting the CAD Structure relations of a new generation of a CAD Document item, can be removed.



Picture 80: The PE_DeleteCADStructure server event method

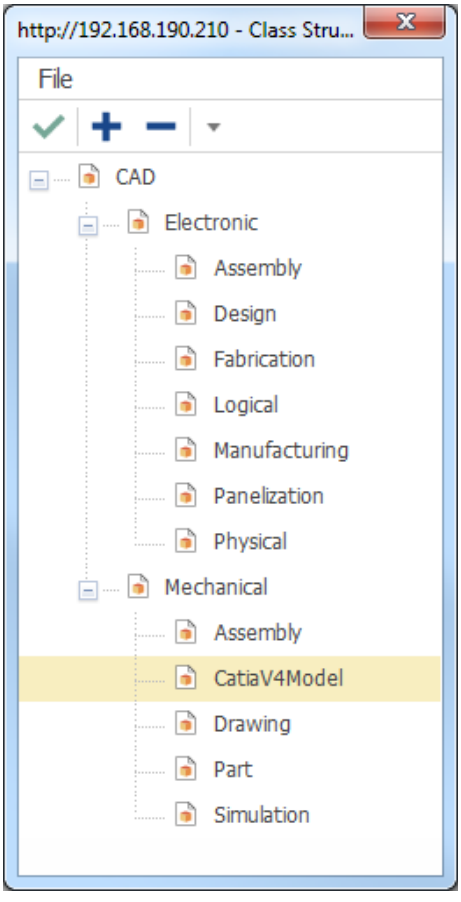
If the server event method “PE_DeleteCADStructure” is removed then the setting “CadStructRelsDeletedAfterVersion” has to be set to “false” in the active PWB Configuration item.

CadStructRelsDeletedAfterVersion	false

Picture 81: Sample CadStructRelsDeletedAfterVersion configuration

Using CATIA V4 models

If structures which contain CATIA V4 models are used a new classification for the item type CAD needs to be defined, for instance “Mechanical/CatiaV4Model”.



Picture 82: Class structure for type ‘CAD’ – CatiaV4Model added

This classification has to be the same as the one defined in the PWBSchema.xml client configuration file, e.g.:

```
<object name="/CAD/Mechanical/CatiaV4Model"
        displayName="NLS_model"
        icon="model">
```

Check for CAD document CATIA release at PDM update

In the Schema file a new functionality can be switched on that asks the user before overwriting a file which has been created with a lower release of CATIA V5.

Example:

```
<checkAuthoringToolVersion value="true" />
```

Default value: false

Optional. Possible values: “true”, or “false”.

Local Workspace Information

It is possible to check the status of the CATIA documents which are downloaded to the local working directory (PWB_XMAP). A list displays the local files and information about their corresponding CAD documents in PDM if they exist.

The icon which opens the window which contains the list of files/documents can be switched on by removing the entry “LocalWorkspace” from the “removeToolbarIcons” definition in the PWBSchema.xml file:

```
<removeToolbarIcons>
      <!-- <icon name="LocalWorkspace" /> -->
      <icon name="Register" />
      <icon name="Synchronize" />
      <icon name="NewPwbWindow" />
      <icon name="DynamicActions" />
</removeToolbarIcons>
```

The PDM attributes which are shown can be configured in the PWBSchema.xml file.

This is an example configuration:

```
<form name="LocalWorkspace" defaultSortAttribute="filename">
      <formAttribute name="modified" visibleLength="8" required="false"
                     listViewRelevant="true" />
      <formAttribute name="filename" visibleLength="10" required="false"
                     listViewRelevant="true" />
      <formAttribute name="item_number" visibleLength="10"
                     required="false" listViewRelevant="true" />
      <formAttribute name="major_rev" visibleLength="6" required="false"
                     listViewRelevant="true" />
      <formAttribute name="generation" visibleLength="6" required="false"
                     listViewRelevant="true" />
      <formAttribute name="name" visibleLength="10" required="false"
                     listViewRelevant="true" />
```

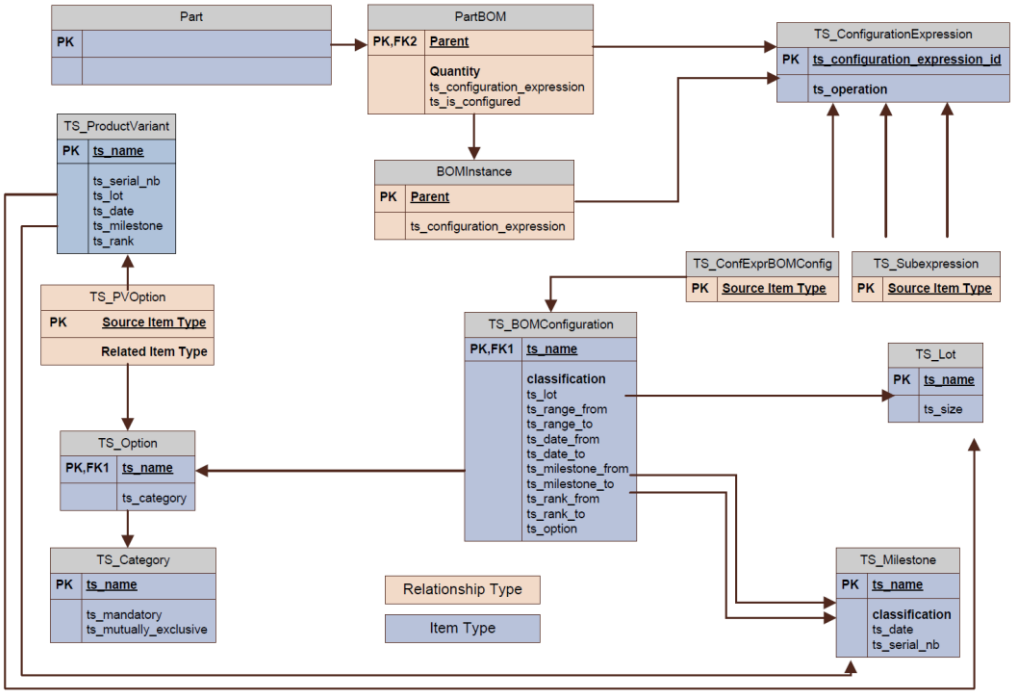
```
        <formAttribute name="state" visibleLength="10" required="false"
                       listViewRelevant="true" />

        <formAttribute name="description" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="created_on" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="modified_on" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="created_by_id" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="modified_by_id" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="locked_by_id" visibleLength="10"
                       required="false" listViewRelevant="true" />

</form>
```

Configuration of BOM Part Structure

In the BOM part structure mode it is possible to create product configurations where, depending of the currently set configuration context, only a sub-set of the product structure is expanded and loaded. With this functionality it is possible to create and to work on different configurations of the same product.

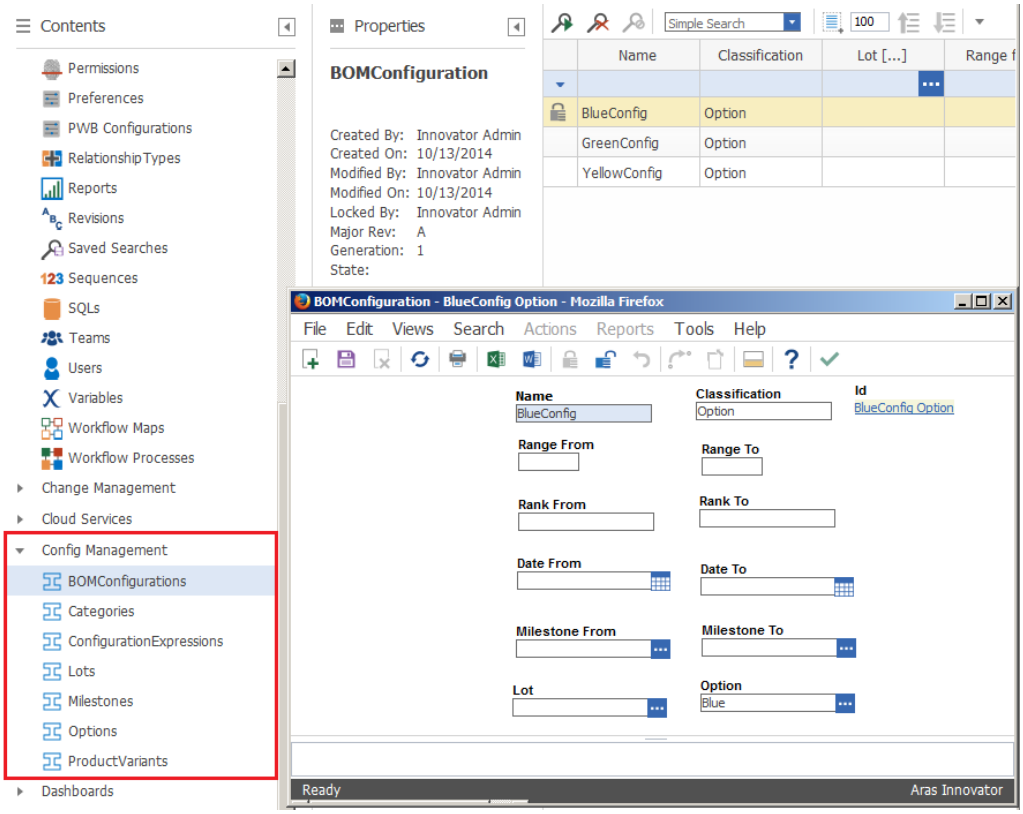
Using the BOM part structure configuration costs additional license fees.

In addition to the three standard PDM Workbench packages which have to be imported with the Aras Innovator import tool (ArasPwbDM, ArasPwbDM_PLM, and ArasPwbDM_Core), two additional packages have to be imported: BomConfig and BomConfig_PLM. These two packages contain the item types which are used by the BOM configuration management functionality:



Picture 83: BOM Configuration Management Type Overview

Importing the packages will enable the administrator to access the types in the Aras Innovator web client and to use them to create different product configurations.



Picture 84: BOM Configuration Management Types in Aras Innovator

Possibility to call a server method for a PDM item

It is possible to call custom server methods with a PDM item and optionally with a dialog as input.

The “PWBSchema.xml” file has to contain the line

```
<customContextAction name=”PwbCustomServerMethod”
                     usedIn=”PdmWindow|QueryDialog”
                     confirm=”true”
                     multiple=”false”
                     dialog=”Query” />
```

after the <contextAction ... /> definitions of an object.

The ‘dialog’ tag is optional and refers to a ‘form’ definition of the same ‘object’ in the PWBSchema.xml file.

The ‘confirm’ and the ‘multiple’ tags are also optional, and they make it possible to not show the confirmation dialog to the user (‘confirm’ = ‘false’) and to pass all items of a multi-selection to the method at once, and only call the method once (‘multiple’ = ‘true’).

If the file “PWBSchemaDisplayNames_Aras_Aras.CATNls” contains the NLS string for the server method, for instance

```
PwbCustomServerMethod = "Custom Server Method";
```

Then the NLS name is shown to the user.

The code of an example server method is shown here (‘multiple’ = ‘false’):

```
ArasUtil ArasUtilObj = new ArasUtil();
Innovator InnovatorObj = this.getInnovator();

string PdmType = this.getProperty("Type");
string PdmKey = this.getProperty("Id");
string PdmClassification = this.getProperty("Classification");

string InputDialogStr = this.getProperty("DialogAttributes");

// Dialog attributes
IDictionary<string, string> InputDialogDict = null;

if (!ArasUtilObj.IsNullOrEmpty(InputDialogStr))
{
       InputDialogDict =
          ArasUtilObj.DialogAttrsStringToDictionary(InputDialogStr);
}

string OutputLogInfo = "";

OutputLogInfo += "PdmType:'" + PdmType + "'" + "|";
OutputLogInfo += "PdmKey:'" + PdmKey + "'" + "|";
OutputLogInfo += "PdmClassification:'" + PdmClassification + "'" + "|";

string OutputUserInfo = "";

OutputUserInfo += "PdmType:'" + PdmType + "'" + "|";
OutputUserInfo += "PdmKey:'" + PdmKey + "'" + "|";
OutputUserInfo += "PdmClassification:'" + PdmClassification + "'" + "|";

if (InputDialogDict != null)
{
       foreach (KeyValuePair<string, string> Attribute in InputDialogDict)
       {
             OutputUserInfo += "'" + Attribute.Key + "' -> '" +
                               Attribute.Value + "'" + "|";
       }
}

Item ItemObject = InnovatorObj.getItemById(PdmType, PdmKey);

// Example action
// ItemObject.unlockItem();

OutputLogInfo += "done";
OutputUserInfo += "done" + "|";
OutputUserInfo += " " + "|";

// Preparing the output
IDictionary<string, string> OutputInfoDict = new Dictionary<string, string>();

OutputInfoDict.Add("LogLines", OutputLogInfo);
OutputInfoDict.Add("UserLines", OutputUserInfo);

Item OutputInfoItem = ArasUtilObj.DialogAttrsDictionaryToItem(InnovatorObj,
OutputInfoDict);

return OutputInfoItem;
```

This is example code for (‘multiple’ = ‘true’):

```
ArasUtil ArasUtilObj = new ArasUtil();
Innovator InnovatorObj = this.getInnovator();

string PdmType = this.getProperty("Type");
string PdmKey = this.getProperty("Id");
string PdmClassification = this.getProperty("Classification");
string DialogAttrs = this.getProperty("DialogAttributes");

string OutputLogInfo = "";

OutputLogInfo += "PdmType:'" + PdmType + "'" + "|";
OutputLogInfo += "PdmKey:'" + PdmKey + "'" + "|";
OutputLogInfo += "PdmClassification:'" + PdmClassification + "'" + "|";
OutputLogInfo += "DialogAttrs:'" + DialogAttrs + "'" + "|";
```

```
string OutputUserInfo = "";

List<string> PdmTypeList = ArasUtilObj.StringToList(PdmType);
List<string>  PdmKeyList = ArasUtilObj.StringToList(PdmKey);
List<string> PdmClassificationList = ArasUtilObj.StringToList(PdmClassification);
List<string> DialogAttrsList = ArasUtilObj.StringToList(DialogAttrs);

for (int i = 0; i<PdmKeyList.Count; ++i)
{
       string CurrentPdmType = PdmTypeList[i];
       string CurrentPdmKey = PdmKeyList[i];
       string CurrentPdmClassification = PdmClassificationList[i];
       string CurrentDialogAttrs = DialogAttrsList[i];

       OutputUserInfo += "CurrentPdmType:'" + CurrentPdmType + "'" + "|";
       OutputUserInfo += "CurrentPdmKey:'" + CurrentPdmKey + "'" + "|";
       OutputUserInfo += "CurrentPdmClassification:'"
                              + CurrentPdmClassification + "'" + "|";
       OutputUserInfo +=
              "CurrentDialogAttrs:'" + CurrentDialogAttrs + "'" + "|";

       // Dialog attributes
       IDictionary<string, string> InputDialogDict = null;

       if (!ArasUtilObj.IsNullOrEmpty(CurrentDialogAttrs))
       {
             InputDialogDict =
                      ArasUtilObj.DialogAttrsStringToDictionary(
                             CurrentDialogAttrs);
       }

       if (InputDialogDict != null)
       {
             foreach (KeyValuePair<string, string> Attribute
                      in InputDialogDict)
             {
                    OutputUserInfo += "'" + Attribute.Key + "' -> '"
                                          +  Attribute.Value + "'" + "|";
             }
       }

       Item ItemObject =
              InnovatorObj.getItemById(CurrentPdmType, CurrentPdmKey);

       // Example action
       //ItemObject.lockItem();
       //ItemObject.unlockItem();

       OutputUserInfo += "done with item " + i + "|";
       OutputUserInfo += " " + "|";
}

OutputLogInfo += "done";
OutputUserInfo += " " + "|";

// Preparing the output
IDictionary<string, string> OutputInfoDict = new Dictionary<string, string>();

OutputInfoDict.Add("LogLines", OutputLogInfo);
OutputInfoDict.Add("UserLines", OutputUserInfo);

Item OutputInfoItem =
       ArasUtilObj.DialogAttrsDictionaryToItem(
              InnovatorObj, OutputInfoDict);

return OutputInfoItem;
```

Part CAD Filter

It is possible to define an attribute on the CAD document item type which defines the so-called “rep type”, that is, the representation type. Only CAD documents with the default rep types will be loaded to the CATIA session.

This functionality only applies to the part structure mode.

Configuration

The server-side configuration settings ‘RepTypeAttribute’, ‘Default3DrepType’, and ‘Default2DrepType’ have to be set.

‘RepTypeAttribute’ has to be set to the internal attribute name of a list attribute which is defined on the type ‘CAD’. The list should contain all the values that are possible for the rep type attribute.

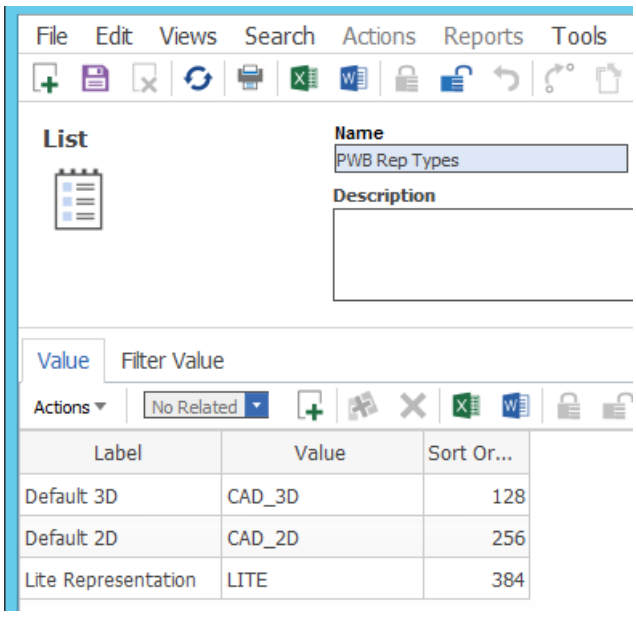
permission_id		Item	Permission
pwb_rep_type	pwb_rep_type	List	PWB Rep Types
release_date	Release Date	Date	

Picture 85: Example rep type attribute

RepTypeAttribute	pwb_rep_type
Default3DRepType	CAD_3D
Default2DRepType	CAD_2D

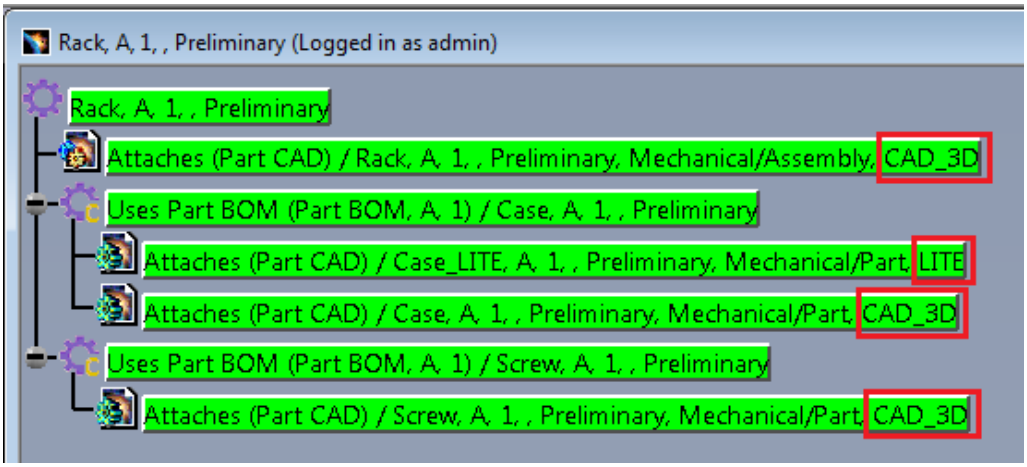
Picture 86: Example rep type value list

The configuration settings ‘Default3DrepType’ and ‘Default2DrepType’ have to be set to the default type for 3D geometry files (CATParts and CATProducts) and 2D geometry files (CATDrawings), respectively.



Picture 87: Sample Part CAD filter configuration

If this functionality is switched on then newly created CAD documents will automatically have the rep type attribute value set to their default value. The value can be changed to a different value (e.g. “LITE”). Only CAD documents with the default rep type value will be loaded.


Rack, A, 1, , Preliminary (Logged in as admin)
Rack, A, 1, , Preliminary
Attaches (Part CAD) / Rack, A, 1, , Preliminary, Mechanical/Assembly, CAD_3D
Uses Part BOM (Part BOM, A, 1) / Case, A, 1, , Preliminary
Attaches (Part CAD) / Case_LITE, A, 1, , Preliminary, Mechanical/Part, LITE
Attaches (Part CAD) / Case, A, 1, , Preliminary, Mechanical/Part, CAD_3D
Uses Part BOM (Part BOM, A, 1) / Screw, A, 1, , Preliminary
Attaches (Part CAD) / Screw, A, 1, , Preliminary, Mechanical/Part, CAD_3D

Picture 88: Different rep type values on CAD documents in a structure

Revising as a different user ('ReviseAs')

By default only administrators can directly promote the life cycle state of an item. The processes in some companies require that also regular designers should be able to promote items. For this it is possible to perform the 'Revise' operation as a different user with the necessary permissions.

Configuration

The server-side configuration setting 'ReviseAs' has to be set to an identity with the necessary permissions, for instance 'Aras PLM':

ReviseAs	Aras PLM

Picture 89: Sample 'ReviseAs' setting

CHAPTER 8

Client Schema File Configuration

This chapter describes the configuration of the client side of the PDM Workbench integration.

Structure of the Schema File

The main purpose of the PDM Workbench Schema File is to define which subset of the objects, relations, and attributes in the PDM system should be made available to the design engineer who is working with CATIA V5 and who needs to save the CATIA files he is working on in a PDM system.

The classes of PDM objects that the user can query, create, etc. will be defined in the Schema file, as well as the dialogs which contain these objects' attributes and the PDM relations which relate the PDM objects to each other.

The Schema file can be edited with a text editor, or a XML editor.

At the root of the Schema XML file, there is the tag "PWBSchemata". Its child tags are named "PWBSchema". The information about every PDM system that can be accessed is defined inside this "PWBSchema" tag. There is one "PWBSchema" tag for every PDM system and every PDM system customization that can be accessed from the PDM Workbench.

```
<!-- root tag -->
<PWBSchemata>
   <!-- out-of-the-box Aras -->
   <PWBSchema system="Aras" customization="Aras"
              displayName="NLS_System" visibleLength="15">
   ...
   </PWBSchema>

   <!-- customization of Aras -->
   <PWBSchema system="Aras" customization="PDM-Customization"
              displayName="NLS_System" visibleLength="15">
   ...
   </PWBSchema>
</PWBSchemata>
```

Attributes of the tag "PWBSchema":

- "system" — Contains the short name of the PDM system. Supported is "Aras" for Aras Innovator.
- "customization" — Contains the name of the customization. If the PDM system is used out of the box without any customization, then the convention is to use the short name as defined for the

"system" attribute.

- "displayName" — Contains the NLS (native language support) name of the PDM system or customization that is defined in the "PWBSchema" XML tag.
- "visibleLength" — Contains the visible length of the display name to be shown in the dialogs of CATIA V5.
- "allowedLength" — Contains the allowed length of the values inserted in the text editor widgets in characters.

NLS Support for Display Names

Many XML tags (PWBSchema, frame, language, object, relation, attribute, etc.) have an attribute with the name "displayName". The string that represents the value of that attribute defines the language-specific display name for that object that the PDM Workbench users can see. The language-specific name is defined in the files "PWBSchemaDisplayNames.CATNls" and "PWBSchemaDisplayNames_SYSTEM_CUSTOMIZATION.CATNls", where SYSTEM is the value of the "system" attribute and CUSTOMIZATION is the value of the "customization" attribute. For system="Aras" and customization="Aras" the name of the CATNls file would be "PWBSchemaDisplayNames_Aras_Aras.CATNls". That file contains the NLS names specific for that PDM system or customization, while "PWBSchemaDisplayNames.CATNls" contains the general definitions that apply to all PDM systems.

In this case, the value for the frame's display name "NLS_UserData" is defined in the file "PWBSchemaDisplayNames_Aras_Aras.CATNls":

File "**PWBSchema.xml**":

```
...
<PWBSchema system="Aras" customization="Aras" displayName="NLS_System"
           visibleLength="15">
...
```

File "**PWBSchemaDisplayNames_Aras_Aras.CATNls**":

```
...
NLS_System = "Aras Innovator";
...
```

Configuration settings

Now the configuration of the PDM Workbench can be defined in the Schema file.

The tags are described in detail in the previous chapter. In this list you can see if the configuration is optional or mandatory.

xmap	optional	see "Exchange map" on page 27
soapTargetUrl	optional	see "SOAP target URL" on page 27
externalSoapClientCallPathEnvVar	optional	see "SOAP client call path environment variable" on page 27
externalFileClientCallPathEnvVar,	mandatory	see "File client call path environment variable" on page 28

addTempPrefix	optional	see “Add Temp configuration” on page 53
pwbWindowColor	optional	see “PWB Window color” on page 28
maxExpansionLevel	optional	see “Maximum expansion level” on page 28
colorSupersededNodesAsOutdated	optional	see “Mark superseded nodes” on page 33
setReadOnly	optional	see “Read only” on page 28
neverReloadCatiaDocAfterUpdate	optional	see “Reload after update” on page 28
checkForCadOwner	optional	see “Check for CAD owner” on page 33
checkFileVersionsInStructure	optional	see “Check file versions in structure” on page 29
checkCatPartsInVisuMode	optional	see “Check CATParts in Visualization mode” on page 29
checkCatiaPartNumbersBeforeUpdate	optional	see “Check CATIA PartNumbers before update” on page 29
allowLoadingDifferentFileVersion	optional	see “Allow loading different file versions” on page 29
downloadDrawingRelated3DFile	optional	see “Download related 3D File” on page 62
showOutOfDateLinkInfo	optional	see “Show out-of-date link info” on page 29
startUpdateAfterCreateNewCatiaFile	optional	see “Update after create new CATIA file” on page 29
uploadChangedFilesAtSyncFailure	optional	see “Upload changed files at sync failure” on page 30
uploadUsingCheckinManager	optional	see “Upload uses Check in Manager” on page 30
checkAuthoringToolVersion	optional	see “Check for CAD document CATIA release at PDM update” on page 68
displayCustomQueryButtons	optional	see “Display custom query buttons” on page 38
requiredDialogAttributes	optional	see “Required attributes” on page 30
queryDialogIsModal	optional	see “Modal dialog” on page 38

checkIfNoQueryAttrsFilled	optional	see “Check if no query attribute filled” on page 38
showDisplayStringInQueryListView	optional	see “Display string in query list view” on page 38
maximumDescriptionAttributeLength	optional	see “Maximum description attribute length” on page 30
showDataSourceNlsValues	optional	see “Show data source NLS values” on page 30
showCreateVersionAtUpdate	optional	see “Create version button” on page 39
defaultCreateVersionAtUpdateValue	optional	see “Default create version value” on page 40
showCreatePartsAtUpdate	optional	see “Show create parts” on page 40
defaultCreatePartsAtUpdateValue	optional	see “Default create parts value” on page 41
renameExistingExMapFilesAtLoad	optional	see “Rename existing files in exchange map” on page 31
duplicateRelatedDrawings	optional	see “Duplicate related drawings” on page 62
removeToolbarIcons	optional	see “Toolbar configuration” on page 34
catiaEnvironment	optional	see “Customer-Specific Environment” on page 34
autoLoginOnSessionExpired	optional	see “Auto login on session expired” on page 31
sessionSettings	optional	see “Session settings” on page 31
installedLanguages	mandatory	see “Language settings” on page 31
dateFormat	mandatory	see “Date format” on page 32
lastModificationDateAttribute	mandatory	see “Internal attribute, do not change. Last modification date attribute” on page 33
standardPartUserDefPropAttribute	optional	see “Standard Part functionality for BOM Part Structure data model” on page 43
partClasses	optional	see “Part classes” on page 33
standardPartFiles	optional	see “Standard Part Functionality” on page 43
boundingBoxAttributes	optional	see “Bounding box definition” on

		page 33
catiaNodeBehaviorDefinitions	optional	see “Configurable CATIA components support” on page 66
templateFiles	optional	see “Management of CATIA templates” on page 47
updateCatiaUserDefinedProperties	optional	see “CATIA user defined attributes” on page 60
updateCatiaLinksInPdm	optional	see “Basic Drawing Link Support” on page 62 and “Basic Multi-Model Link Support” on page 63
attrsModifiableAtRevise	optional	see “Modifiable attributes” on page 41
expandStructureForRevise	optional	see “Expand structure” on page 41

"object": 1 - n

This tag contains the definition of a PDM object class which can be used (queried, created, etc.) by the user.

The definition of PDM object classes, their corresponding dialogs and the actions that can be performed on them are described in the chapter **PDM Objects**.

"relation": 1 - n

This tag contains the definition of PDM relation classes that can be accessed (expanded, created, etc.) by the user.

The definition of PDM relation classes, their corresponding dialogs and the actions that can be performed on them are described in the chapter **PDM Relations**.

"attribute": 0 - n

The definition of PDM attributes that are referenced in dialogs.

Attributes and dialog forms are explained in the chapter **PDM Attributes and Form Attributes**.

"pwbAttribute": 0 - n

The definition of attributes that do not correspond directly to PDM attributes of PDM objects.

Attributes and dialog forms are explained in the chapter **PDM Attributes and Form Attributes**.

"dataSource": 0 - n

Data sources contain attribute values. By assigning data sources to attributes default values for these attributes can be defined.

Data sources are explained in the chapter **Data Sources**.

PDM Attributes and Form Attributes

Every PDM attribute that is displayed in a dialog form should be defined in an "attribute" tag.

The attribute definition contains the following attributes:

- "name" Mandatory, must correspond to the PDM attribute's name.
- "displayName" Mandatory. As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.
- “dataSource” Optional. The data source includes the possible values for this attribute.
- "isFileName" Optional. If is set to "true" the value of the corresponding input file name is checked about illegal[2] characters when creating a file.
- "isPartNumber” Optional.
- “autoName” Optional.
- "isDerived" Optional.

Example:

```
<attribute name="name" displayName="NLS_Name" isFileName="true"
           isPartNumber="true" autoName="true"/>

<attribute name="current" displayName="NLS_current"
           dataSource="LifeCycleStates"/>

<attribute name="revision" displayName="NLS_Revision" />
```

A form definition contains form attributes which reference the previously defined PDM attribute.

The form attribute definitions contain the following attributes:

- "name" Mandatory, must correspond to the PDM attribute's name.
- "displayName" Optional. As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization. If not defined here the display name of the "attribute" tag will be used.
- "mode" Possible values are "output" (read-only), "update" (can be modified), or “select" (e.g. for combo boxes).

  Default is "output".
- "visibleLength" Optional, the length of the text editor widget in characters.
- "allowedLength" Optional, the length of the value that can be inserted in the text editor widget in characters.
- "required" "true" or "false". If "true", then a value must be set.

[2] Filenames must not contain control characters, non printable characters and any of the following characters: *?:;\/<>|

Default is "false".

- "widgetType" — Possible values are "SingleLineEditor", "MultiLineEditor", "ComboBox", "SingleCheckBox", "CheckBoxes", "RadioButtons", "SingleSelectorList", "MultiSelectorList", "NameValueList", “Date”.

  Default is "SingleLineEditor".

- "embeddedObjAttr" — Optional. If the PDM attribute refers to a different PDM attribute in a contained object attribute, then this XML attribute's value contains that object attribute's name.
- "embeddedAttribute" — Optional. The name of the PDM attribute of the embedded object. If "embeddedObjAttr" is set, then "embeddedAttribute" must be set, too.
- “dataSource” — Optional. The value defines the link to a data source that is more special than the linked data source in the <attribute> tag.
- "listViewRelevant" — "true" or "false". If "true", then the attribute will appear in the query list view.

  This attribute should only be added to "Query" forms. Please refer to the *PDM Workbench User Manual* for more information.

  Default is "false".

- "sort" — Possible values are "true" or "false". If "true", then the attribute can be sorted in the query result window.
- "displayOnly" — Possible values are "true" or "false". If "true", then the display value of the value of the data source will be used.
- "entryAllowed" — Possible values are "true" or "false". If "true", then the user can enter a text additional to the attached data source.

Example:

```
<form name="Query">
   <formAttribute name="name" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />
```

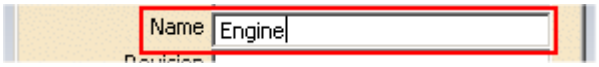
Form definitions generally refer to classes of PDM objects (query form, properties form, etc.). The definition of PDM object classes is described in chapter **PDM Objects**.

Description of the Widget Types

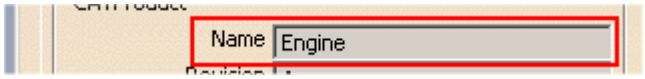
There are ten different widget types available to build up dialogs with. All widget types except “SingleLineEditor”, “MultiLineEditor” and one mode of “NameValueList” can only be used on attributes that have certain kinds of Data Sources attached. Data Sources are a container of a limited set of values.

The detailed explanation of Data Sources you can find in chapter **Data Sources**.

SingleLineEditor Supports "update" and "output" mode.
Can be used for attributes with no data source attached and also for attributes with data sources of type “SingleValue”.

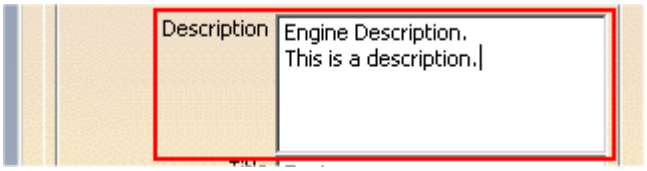


Picture 90: Single Line Editor Widget, update mode



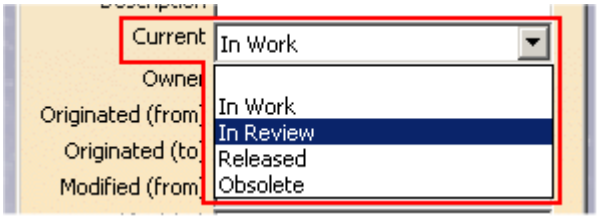
Picture 91: Single Line Editor Widget, output mode

MultiLineEditor Supports "update" and "output" mode.
Can be used for attributes with no data source attached and also for attributes with data sources of type “ValueList”.



Picture 92: Multi Line Editor Widget, update mode

ComboBox Supports “select” and “output” mode.
This widget type can only be used for attributes with data sources of type “ValueList”, “BooleanValueList” or “invokeMessage” if this message returns a set of values.

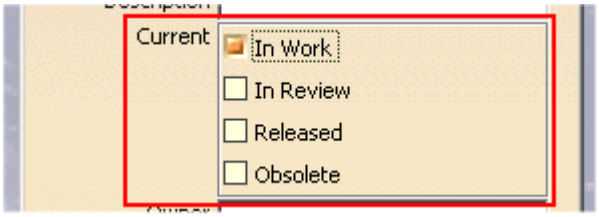


Picture 93: Combo Box Widget, select mode

SingleCheckBox Supports “select” and “output” mode.
Needs an attribute with a data source of type "BooleanValueList".
This widget should be used only for required attributes or for attributes that are only displayed, already set to a value and cannot be updated.

Picture 94: Single Check Box Widget, select mode

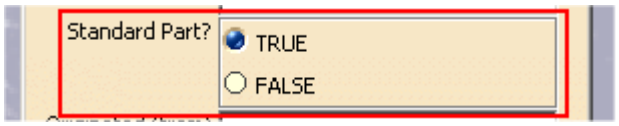
CheckBoxes Supports “select” and “output” mode.
Possible for attributes with data sources which contain several values (type "ValueList”, “BooleanValueList” or “invokeMessage”) and where the user can select more than one value.



Picture 95: Check Boxes Widget, select mode

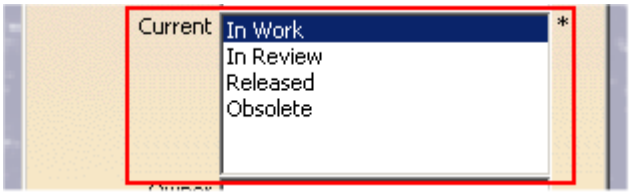
RadioButtons Supports “select” and “output” mode.
Possible for attributes with a data source of type "ValueList", “BooleanValueList” or “invokeMessage” where only one value can

be selected at one time.
They should be used for required attributes only, because one value has always to be selected.



Picture 96: Radio Buttons Widget, select mode

SingleSelectorList Supports “select” and “output” mode.
It represents a list with one column where one item is selectable.
It can be used for all attributes with attached data sources of type "ValueList", “BooleanValueList” or “invokeMessage”.



Picture 97: Single Selector List Widget, select mode

MultiSelectorList Supports “select” and “output” mode.
It represents a list with one column where several items are selectable.
(The multi-selector list looks like the single-selector list, except that more than one item of the list can be selected.)
It can be used for all attributes with attached data sources of type "ValueList", “BooleanValueList” or “invokeMessage”.

NameValueList Supports “select”, “update” and “output” mode.
It represents a list with two columns (e.g. for name value sets)

→ when working in update mode the widget type can be used for all attributes within the Schema file (no data source needed). The user can change every single column item (columns can be empty).

→ when working in select mode the widget type can only be used for attributes with data sources of type “NameValueList” attached. (widget acts as a filter then)

Login Form

This tag contains a description of the Login form. It defines the attributes needed for logging in to the PDM system. Generally it contains the attributes "login name", "password", and “database” at least, though other attributes like "group" can be defined if it is necessary for the PDM system.

Example:

```
<form name="Login" info="ShowOnlyLoginData">

   <frame displayName="NLS_UserData">

      <pwbFormAttribute name="PWBLoginUser" widgetType="ComboBox"
                        mode="update" visibleLength="10" required="true"
                        entryAllowed="true" dataSource="UserNames" />

      <pwbFormAttribute name="PWBLoginPassword"
                        widgetType="SingleLineEditor" mode="update"
                        visibleLength="10" required="false" />
```

```
      <formAttribute name="LoginDatabase" widgetType="ComboBox"
                     mode="update" visibleLength="10" required="true"
                     entryAllowed="false" />
   </frame>
</form>
```

The XML tags inside the "frame" tag describe how the attributes "user" and "password" are displayed in the login dialog.

Mandatory.

PDM Objects

Object XML tags define PDM object classes that can be used in the PDM Workbench application. They represent the subset of objects defined within the PDM system which are needed in a PDM-CAD integration.

An "object" XML tag contains the following attributes:

- "name" The internal PDM class name.
- "displayName" The class name that is shown to the user.
- "icon" The icon that represents the class in the PDM window and the list window.

Example:

```
<object name="/Part/Assembly" displayName="NLS_Assembly" icon="Aras_Part">
```

For the icon to be displayed correctly in the PDM window a bitmap file with the name of the icon (in this example "Aras_Part.bmp") must exist in the subdirectory "resources\graphic\icons\normal" of the CATIA V5 directory (e.g. "intel_a" on Windows 32 Bit CATIA installation, and “win_b64” on Windows 64 Bit CATIA installation).

The list view window needs a bitmap file with the file name (icon name) + "16x16.bmp", in this example "Aras_Part16x16.bmp".

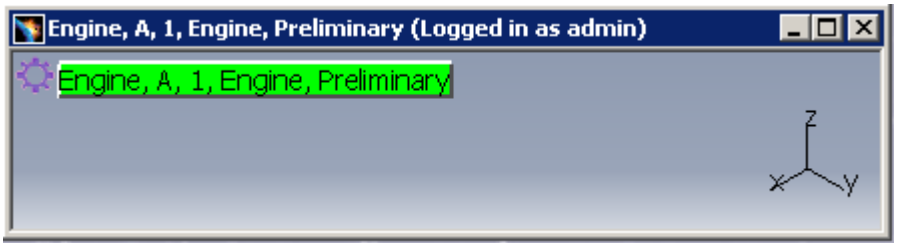
As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.

Description of PDM Objects

The "description" tag defines which of the attributes of the class should be displayed beside the icon. In this example, these are the attributes “item_number”, “major_rev”, “generation”, “name”, and “state”.

Example:

```
<description>
   <descAttribute name="item_number" />
   <descAttribute name="major_rev" />
   <descAttribute name="generation" />
   <descAttribute name="name" />
   <descAttribute name="state" />
</description>
```



Picture 98: PDM Node in PWB window

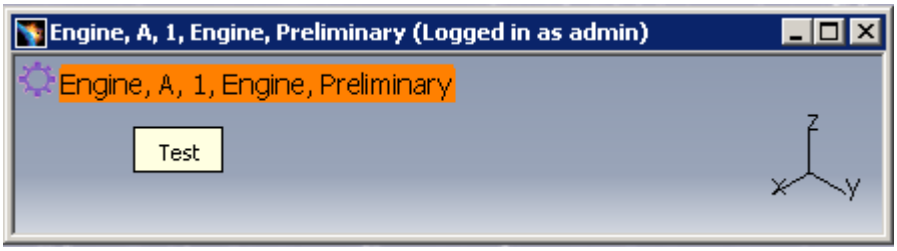
Tooltip of PDM Objects

The "tooltipAttribute" tag defines which string has to be shown as tooltip for the object. The value can be the name of an attribute of the object. If this value is empty then the description defined in the chapter above will be used.

Example:

```
<tooltipAttribute name="description" />
```

The attribute “description” has the value “Test”.



Picture 99: Tooltip of PDM Node in PWB window

Actions on PDM Objects

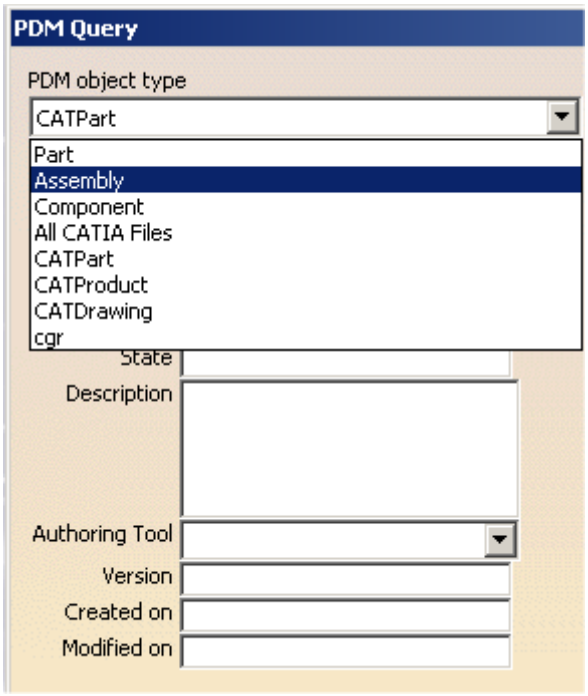
Actions that can be performed with PDM objects have to be defined in the Schema file. There are two kinds of actions: So-called toolbar actions, which are started by clicking on an icon in the PDM Workbench toolbar, and context actions, which are started by right-clicking on the node and selecting one of the context menu items.

Toolbar actions are defined with an "action" tag. The action "Query" can be defined on any object type.

Example:

```
<!-- * all PWB toolbar actions permitted for this object * -->
<action name="Query" />
```

If, for instance, the action "Query" is defined for the object type "/Part/Assembly", then, when the user clicks on the "Query" toolbar icon, the type "Assembly" (display name) is included in the query dialog list, otherwise it is not.



Picture 100: Select PDM object type in “PDM Query” dialog

Context Actions

Context actions are declared similarly to toolbar actions. For context actions, the tag "contextAction" is used.

Example:

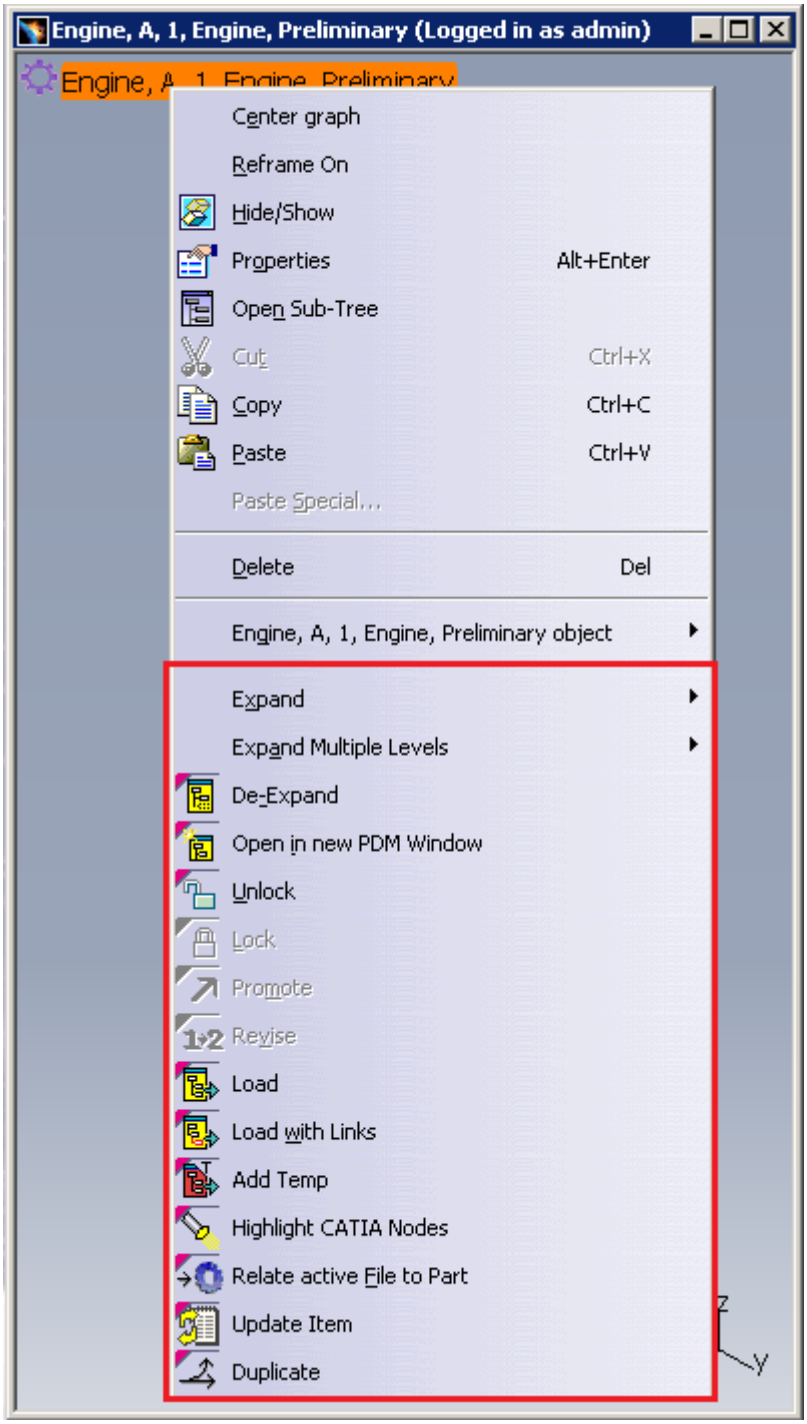
```
<!-- * all PWB context actions permitted for this object * -->
<contextAction name="Expand" usedIn="PdmWindow" />
<contextAction name="MultipleExpand" usedIn="PdmWindow" />
<contextAction name="DeExpand" usedIn="PdmWindow" />
<contextAction name="CheckIn" usedIn="PdmWindow|QueryDialog" />
<contextAction name="CheckOut" usedIn="PdmWindow|QueryDialog" />
<contextAction name="Promote" usedIn="PdmWindow|QueryDialog" />
<contextAction name="Revise" usedIn="PdmWindow|QueryDialog" />
<contextAction name="LoadStructure" usedIn="PdmWindow" />
<contextAction name="AddTemp" usedIn="PdmWindow" />
<contextAction name="Highlight" usedIn="PdmWindow" />
<contextAction name="UpdateSubCmpRels" usedIn="PdmWindow|QueryDialog" />
<contextAction name="UpdateParentCmpRel" usedIn="PdmWindow|QueryDialog" />
<contextAction name="UpdateItem" usedIn="PdmWindow" />
<contextAction name="DeleteItem" usedIn="PdmWindow" />
<contextAction name="DeleteNewestVerstion" usedIn="PdmWindow" />
```

The attribute "usedIn" defines where the context action is shown. The attribute value "PdmWindow" activates the action in the PDM window. The attribute value "QueryDialog" activates the action in the result list view of the query dialog. These two attribute values can be combined with the pipe "|".

The context action “Properties” is default and not explicitly defined in the Schema file.

The context action "Properties" has a window with two tabs for “Properties”, and "UpdateItem". The corresponding dialog forms must also be defined for that class.

If a certain context action is defined for a PDM object class in the Schema file, then the corresponding context menu entry for object nodes of that class exists.



Picture 101: Context actions for the type /Part/Assembly

Some things to keep in mind regarding the definition of context actions:

If the context action "Unlock" (CheckIn) is defined, then the dialog forms "CheckInNew" and "CheckIn" must also be defined.

If the context action "Lock" (CheckOut) is defined, then the dialog form "CheckOut" must also be defined.

The context action “Load” (LoadStructure) should only be defined on part objects.

Disabling context menu items in the CATIA structure window

In the corresponding “contextAction” definition the XML attribute “removeFromCatiaWindow” needs to be set to “true”:

```
<contextAction name="CheckIn" usedIn="PdmWindow|QueryDialog"
               removeFromCatiaWindow="true" />
```

If that is done the corresponding context action will not be available in the CATIA window anymore. It may still be available as a context menu in the PDM structure window or as a context menu in the query dialog result list.

PDM Object Forms

The following forms can be defined for an object class:

"Query", "Properties", "UpdateItem", "CheckInNew", "CheckIn", "CheckOut"

The "CheckIn" and "CheckOut" forms do not have to contain any attributes, they just need to be defined.

Example:

```
<form name="Query">

   <formAttribute name="item_number" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="major_rev" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="generation" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="name" widgetType="SingleLineEditor" mode="update"
                  visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="state" widgetType="SingleLineEditor" mode="update"
                  visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="unit" widgetType="ComboBox" mode="update"
                  visibleLength="15" required="false" />

   <formAttribute name="make_buy" widgetType="ComboBox" mode="update"
                  visibleLength="15" required="false" />

   <formAttribute name="description" widgetType="MultiLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="created_on" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="modified_on" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

</form>
```

PDM Relations

Relation XML tags define PDM relation classes that can be used in the PDM Workbench application. They represent the subset of the relations defined in the PDM system that are needed in PDM-CAD integration.

As with the "object" XML tags, a "relation" XML tag contains the following attributes:

- "name" The internal PDM class name.
- "displayName" The class name that is shown to the user.
- "icon" The icon that represents the class in the PDM window and the list window.
- "createAllowed" Defines whether relations of this class can be created by the user. Relations are created by copying and pasting object nodes. If "createAllowed" is "true", then, at paste, the relation class is included in the list of applicable relations, otherwise it is not. Please refer to the *PDM Workbench User Manual* for more information.
- "expandAllowed" Defines if it is allowed to expand this relation class explicitly in the context action "Expand".

Example:

```
<relation name="Part BOM" displayName="NLS_PartBOM"
          icon="Aras_Relation" createAllowed="true"
          expandAllowed="true">
```

As for the PDM object classes, a bitmap file for the icon must exist in the CATIA icon subdirectory ("IconName.bmp" and "IconName16x16.bmp").

As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.

Description of PDM Relations

As with PDM object definitions, the "description" tag defines which of the attributes of the class should be displayed beside the icon.

In this example the attributes "Class", "major_rev", and "generation" are defined as the description.

Example:

```
<description>
   <descAttribute name="Class" />
   <descAttribute name="major_rev" />
   <descAttribute name="generation" />
</description>
```

"Relationship" tags

The tags "leftToRightRelationship" and "rightToLeftRelationship" define the relationships for the two sides of the relation. They contain the following XML attributes:

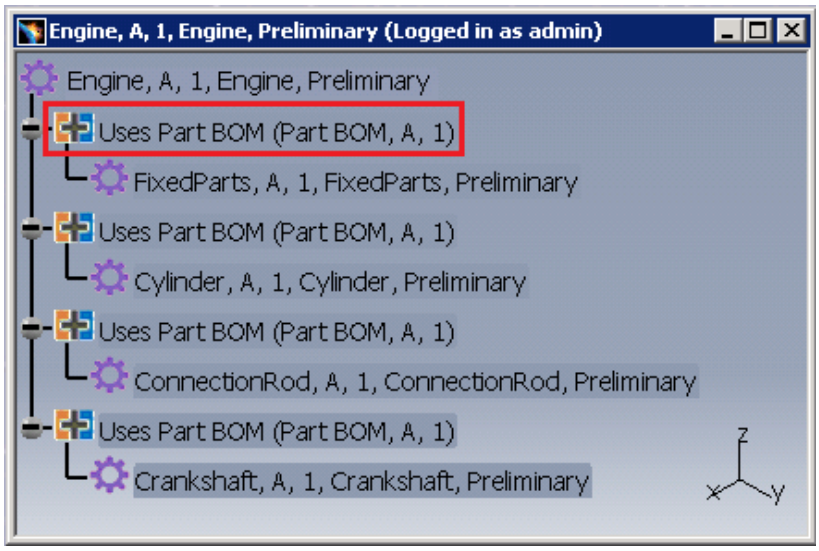
- "name" The internal PDM relationship name.
- "displayName" The relationship name that is shown to the user.
- "multipleExpand" If "multipleExpand" is "true", then the relationship appears in the "Expand Multiple Levels" context menu, otherwise it does not. For this the context action "MultipleExpand" must be defined on the PDM object. The default value is "false".

Example:

The Aras relation "Part BOM" has the relationships "Part BOM_LeftToRight" (display name is "Uses Part BOM") and "Part BOM_RightToLeft" (display name is "Is Used in Part BOM").

```
<leftToRightRelationship name="Part BOM_LeftToRight"
                         displayName="NLS_PartBOM_LeftToRight"
                         multipleExpand="true" />
<rightToLeftRelationship name="Part BOM_RightToLeft"
                         displayName="NLS_PartBOM_RightToLeft"
                         multipleExpand="true" />
```

The text that describes a relation icon is the display name of the expanded relationship ("Part BOM“ in this example) as well as the parameters defined in the "description" XML tag in parentheses “Class”, “major_rev”, and “generation” in this example).


Engine, A, 1, Engine, Preliminary (Logged in as admin)
Engine, A, 1, Engine, Preliminary
Uses Part BOM (Part BOM, A, 1)
FixedParts, A, 1, FixedParts, Preliminary
Uses Part BOM (Part BOM, A, 1)
Cylinder, A, 1, Cylinder, Preliminary
Uses Part BOM (Part BOM, A, 1)
ConnectionRod, A, 1, ConnectionRod, Preliminary
Uses Part BOM (Part BOM, A, 1)
Crankshaft, A, 1, Crankshaft, Preliminary
Z
X
Y

Picture 102: Relation icon with relationship and description attribute

“Left and Right Object” Classes

The tags "leftObject" and "rightObject" define which object classes are valid for this relation.

In the following example the relation is between CATIA parts:

```
<leftObject name="/Part/Assembly" />
<rightObject name="/Part/Assembly" />
<rightObject name="/Part/Component" />
```

PDM Relation Forms

The following forms can be defined for a relation class:

"Properties", "UpdateItem", "Create"

The definition of the dialog forms for relations is similar to the definition of the dialog forms for objects.

Example:

```
<form name="UpdateItem">
   <formAttribute name="reference_designator"
                  widgetType="SingleLineEditor" mode="update"
                  visibleLength="15" />
</form>
```

Data Sources

Data sources describe a static set of values that are already known when writing the Schema file. The set of these values will never change during the lifetime of the PDM Workbench.

Data Source "Value" tag

The value tag of static data sources contains the following XML tags:

- "name" The PDM name of the attribute.
- "displayName" The dialog display name of the attribute.
- "booleanValue" "true" or "false" to assign the correct value to the attribute names (this tag is only used for type BooleanValueList).
- "valueName" The PDM name of the value attribute (this tag is only used for type NameValueList).
- "displayValue" The dialog display name of the value attribute (this tag is only used for type NameValueList).

Static data sources can be of type:

SingleValue: the data source contains only one static element.

Example:

```
<dataSource name="Autoname" type="SingleValue">
   <value name="autoname" displayName="NLS_Autoname" />
</dataSource>
```

ValueList: the data source contains a set of static value elements.

Example:

```
<dataSource name="LifeCycleStates" type="ValueList">
   <value name="Preliminary" displayName="NLS_Preliminary"/>
   <value name="Review" displayName="NLS_Review"/>
   <value name="Approve" displayName="NLS_Approve"/>
   <value name="Released" displayName="NLS_Released"/>
</dataSource>
```

BooleanValueList: the data source contains exactly the value pair "true" and "false".

Example:

```
<dataSource name="PlusOrMinus" type="BooleanValueList">
   <value name="+" displayName="NLS_ValueSetPlus" booleanValue="true" />
   <value name="-" displayName="NLS_ValueSetMinus" booleanValue="false" />
</dataSource>
```

NameValueList: the data source contains a list of name-value pairs.

Example:

```
<dataSource name="Test" type="NameValueList">
```

```
   <value name="test1" displayName="NLS_test1" valueName="test1_value"
          displayValue="NLS_test1_value" />
   <value name="test2" displayName="NLS_test2" valueName="test2_value"
          displayValue="NLS_test2_value" />
</dataSource>
```

Complete example of using a data source tag:

The attribute “CheckedOut” can be assigned exactly to “true” or “false”. Within the dialog this will be expressed by showing a “+” or a “-“ sign. Therefore we define a data source called “PlusOrMinus” and attach this container to the attribute description.

```
<attribute name="CheckedOut" displayName="NLS_CheckedOut"
           dataSource="PlusOrMinus" />
<dataSource name="PlusOrMinus" type="BooleanValueList">
   <value name="+" displayName="NLS_ValueSetPlus" booleanValue="true" />
   <value name="-" displayName="NLS_ValueSetMinus" booleanValue="false" />
</dataSource>
```